

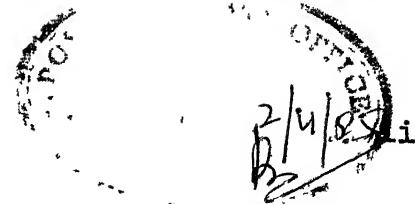
**GOAL PROGRAMMING ALGORITHMS
FOR
NETWORK FLOW PROBLEMS**

**A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY**

**by
A. SRINIVAS**

**to the
INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
APRIL, 1985**

CERTIFICATE



This is to certify that the present work on "Goal Programming Algorithms for Network Flow Problems," by A. Srinivas has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

RK Ahuja

(R. K. Ahuja)

Lecturer

Industrial and Management Engg. Program

Indian Institute of Technology

Kanpur 208 016, INDIA

April, 1985

12 JUN 1985

111 KAMALI
CENTRAL
87470

IMEP-1985-M-SRI-GOA

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Dr. R.K. Ahuja for his valuable suggestions and excellent guidance throughout the span of this work.

I am also thankful to my friend Mr. V.V.S.N. Murthy for his help during the course of this work.

Lastly, I thank Mr. J.K. Misra for his excellent typing of this manuscript, and Mr. Buddhi Ram Kandiyal for his neat cyclostyling work.

A. Srinivas

CONTENTS

<u>Chapter</u>		<u>Page</u>
I.	INTRODUCTION	
	1.1 Introduction	1
	1.2 Outline of the Thesis	3
II.	PRELIMINARIES	
	2.1 Introduction	7
	2.2 Graph Theory Notations	7
	2.3 Literature Review	9
	2.4 Bicriteria Linear Programming	11
	2.5 An Overview of Goal Programming	20
III.	NETWORK GOAL PROGRAMMING ALGORITHMS	
	3.1 Introduction	27
	3.2 Notations	29
	3.3 Tracing the Trade-off Curves	29
	3.4 Mathematical Formulation of WGNF Problem	36
	3.5 Development of the Algorithm	37
	3.6 Statement of Algorithm	53
	3.7 Numerical Example	58
	3.8 Mathematical Formulation of IGFN Problem	59
	3.9 Algorithm for the IGFN Problem	64
	3.10 Numerical Example	67
	3.11 Computational Results	67
	REFERENCES	77
	APPENDIX	

ABSTRACT

In this dissertation, we consider the weighted goal programming and interval goal programming problems in the network context. Special structure embedded in their problems is used to develop computationally efficient algorithms. In weighted goal network flow [WGNF] problem, the decision maker specifies aspiration levels for each of the objectives and weighting factors for each of the deviations from the aspiration levels. On the other hand, in interval goal network flow [IGNF] problem, the decision maker provides a range of aspiration levels for each objective and weighting factors for each of the deviations from the specified range of aspiration levels. Both the algorithms developed trace a path in the feasible region and obtain the solutions which minimize the sum of weighted deviations. The WGNF and the IGFN algorithms are based on the parametric approach and fully exploit the special structure of the minimum cost flow problem in order to perform all the computations on the network itself. Computer programs were written for both the algorithms, and tested on randomly generated network problems. Results of their investigations are presented.

CHAPTER I

INTRODUCTION

1.1 Introduction:

Operations Research is a scientific approach to solve complex problems arising in the management of large systems of men, machines, materials and money. Operations Research is a decision science which helps management to make better decisions. Network model is a branch of study in Operations Research. Due to the wide applicability of network models in real world, it is considered as an important branch of study in Operations Research. In the present day, we find that complex intriguing problems arising in production-distribution systems, military logistics systems, urban traffic systems, railway systems, communication systems, pipeline network systems, facilities location systems, file merge systems, electrical networks etc. can be tackled by constructing an appropriate network model. Furthermore, network geometry (or relationships) can be easily displayed in two-dimensional drawings, greatly simplifying the communication problem between the analyst and the client for whom the model is designed. In the recent years, network flow problems have received special attention due to significant

advances in implementation technology and solution techniques, thereby increasing the applicability of the network models substantially.

In this dissertation, we consider the bicriteria minimum cost flow problems and propose two algorithms i.e. weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms for obtaining optimum solutions. These models are useful when two conflicting objectives are to be simultaneously considered, i.e. one is interested in minimizing the total cost as well as total time.

One application of the bicriteria minimum cost flow problem arises in a distribution system. Suppose V units of perishable items (i.e. fruits) are to be sent from source to sink along the road network. The two objectives to be considered simultaneously are total cost and total time. We would like to transport the items in such a way that they are not perished before they reach the sink and at the same time the total cost of transportation should be minimum.

Another application of the bicriteria minimum cost flow problem arises in communication networks. Suppose V units of messages/sec. are required to be transmitted from source to sink along a communication network. The decision maker would like to consider two objectives simultaneously. One of the objectives may be to minimize total cost of sending messages,

and the other being the minimization of total amplification cost.

Another important application of bicriteria network flow problem is in the area of flow through a pipeline network. The decision maker would like to minimize the total operating cost for sending the flow as well as minimize the total time to transmit flow from source to sink.

In this work, we develop specializations of goal programming i.e. weighted goal and interval goal network flow algorithms based on parametric approach. The special structure of minimum cost flow problem is exploited to perform all the computations on the network itself. Efficient tree data structures are used to further enhance the efficiency of these computations. Computational investigations with the above approaches are found to be very encouraging and presented in sufficient detail.

1.2 Outline of the Thesis:

In this section, we give a brief outline of this thesis.

In Chapter II, we present some of the preliminary knowledge required to understand the work done in the subsequent chapters. We first review the literature related to the bicriteria minimum cost flow problem. The graph theory notations adopted in this work are also described. Finally, an

overview of the bicriteria network flow problem and the goal programming are briefly outlined.

Chapter III embodies the main work done on WGNF problem and IGFN problem. They can be formulated as shown below:

Weighted Goal Network Flow Problem:

$$\text{Min. } Z = w_1\alpha_1 + w_2\alpha_2 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \\ V, & \forall i \in N \\ 0, & \text{Otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1$$

Interval Goal Network Flow Problem:

$$\text{Min. } Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i = 1 \\ V, & \text{if } i = n, \quad \forall i \in N \\ 0, & \text{Otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_3 - \alpha_3 = c_2$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_4 - \alpha_4 = d_2$$

where,

x_{ij} is the amount of flow on arc (i,j) .

c_{ij} is the cost of unit flow on arc (i,j) .

d_{ij} is the budget required for unit flow on arc (i,j) .

b_{ij} is the capacity of arc (i,j) .

v is the net flow of commodity from source to sink n .

c_i is the aspiration level for cost objective function.

d_i is the aspiration level for budget objective function.

α_i is the positive deviation.

β_i is the negative deviation.

We suggest two exact algorithms to solve bicriteria network flow problem. They are weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms. Both these algorithms are based on the parametric algorithm for constrained minimum cost flow problem [1].

The WGNF algorithm searches for an optimum solution which satisfies the aspiration levels specified for each objective by the decision maker. Whereas IGFN algorithm searches for an optimum that lies in the specified range of aspiration levels for each of the objectives.

The computational investigations of both the algorithms are given at the end of Chapter III. Computer programs are written for both the algorithms and the computational performances are ascertained by solving different sized problems. Efficient data structures are implemented to represent the basis tree which requires comparatively less storage. Data structures based on augmented threaded index method are used and results of the computational investigations are presented.

CHAPTER II

PRELIMINARIES

2.1 Introduction:

In this chapter, we review some of the relevant concepts of the bicriteria minimum cost flow problem and goal programming. The basic reason behind this is to prepare a sufficient background which will help in understanding the algorithms developed in the subsequent chapters.

This chapter is divided into five sections. The graph theory notations adopted in this work are presented in Sec. 2. The literature related to the bicriteria minimum cost flow problem is surveyed in Sec. 3. A brief review of various methods of solving bicriteria minimum cost flow problem and a parametric algorithm for solving constrained minimum cost flow problem [1] are presented in Sec. 4. In Sec. 5 a brief description about goal programming and its importance to management in decision making to real world situations, and various goal programming techniques are discussed.

2.2 Graph Theory Notations:

Some notations and well-known concepts of graph theory that are being used throughout the thesis are mentioned below.

A directed graph $G = (N, A)$, consists of a finite set N of elements, called nodes, and a set A of ordered pairs of nodes called arcs. A directed network is a directed graph in which numerical values are attached to the nodes and arcs of the graph. Let $n = |N|$ and $m = |A|$. The two specified nodes l and n are called the source and the sink respectively.

An arc (i, j) has two end points, i and j , and it is said to be incident from node i and incident to node j . Let $I(i)$ and $O(i)$ denote, respectively, the sets of arcs incident to and incident from node i . The degree of a node i is the number of arcs incident to or incident from that node.

A path in $G = (N, A)$ is a sequence i_1, i_2, \dots, i_r of distinct nodes of N such that either $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$ for each $k = 1, \dots, r-1$. A directed path is defined similarly, except that $(i_k, i_{k+1}) \in A$ for each $k = 1, \dots, r-1$. A cycle is a path together with an arc (i_r, i_1) or (i_1, i_r) . A directed cycle is a directed path together with the arc (i_r, i_1) .

A graph $G = (N', A')$ is a subgraph of $G = (N, A)$ if $N' \subseteq N$ and $A' \subseteq A$. A graph $G = (N', A')$ is a spanning subgraph of $G = (N, A)$ if $N' = N$ and $A' \subseteq A$.

Two nodes i and j are said to be connected if there is atleast one path between them. A graph is said to be

connected if all pairs of nodes are connected, otherwise it is called disconnected. A set $Q \subseteq A$ such that the graph $G = (N, A-Q)$ is disconnected and no subset of Q has this property, is called a cocycle of G . A cocycle is a cutset if it disconnects source and sink.

A graph is acyclic if it does not contain any cycle. A tree is a connected acyclic graph. A subtree of a tree T is a subgraph of T as well as a tree. A tree T is said to be a spanning tree of G if T is a spanning subgraph of G . Arcs belonging to a spanning tree T are called tree-arcs and arcs not belonging to T are called nontree-arcs. A spanning tree of $G = (N, A)$ has exactly $(n-1)$ tree-arcs.

2.3 Literature Review

One of the methods of solving the bicriteria minimum cost flow problem is to consider one of the objectives as a constraint and solve the resultant constrained minimum cost flow [CMCF] problem. We will first present a brief review of the constrained flow problems.

Hultz and Klingman [9] have suggested a partitioning method in conjunction with the simplex method for solving constrained generalized network flow problems. Takashi Kobayashi [12] proposed a primal-dual method for solving CMCF problem. He has associated two dimensional distance for each arc. The first element is related to the cost and the second

one to the coefficient of the additional constraint. This method is suitable in cases when degeneracies often occur.

A parametric algorithm for solving CMCF problem is developed by Ahuja, Batra and Gupta [1]. The algorithm uses the concepts from parametric linear programming and fully exploits the topological structure embedded in the problem. The algorithm can also be used to generate all the efficient points of the bicriteria minimum cost flow problem.

Chen and Saigal [4] have suggested a primal algorithm for solving a capacitated network flow problems with additional linear constraints.

Klingman and Mote [10] have reviewed the fundamental theoretical results for the general multi-criteria linear programming and the relevant exploitable characteristics of the network basis. Two algorithms are then developed for solving the multicriteria network flow problems efficiently. One approach determines the set of all non-dominated solutions to the problem. The other approach is a network variant of the surrogate criterion linear programming approach. Ignizio [7] has suggested a straight forward weighted integer goal programming for generalized networks models for integer programming problems. His method is simplex and robust.

Practically no study was made in the field of application of goal programming techniques to network flow problems. This has motivated us to propose two algorithms for WGNF and IGFN problems.

2.4 Bicriteria Linear Programming:

Bicriteria linear programming deals with the optimisation of two objective functions (multicriteria deals with two or more than two objective functions) simultaneously. A decision situation is generally characterised by multiple objectives. Some of these objectives may be complementary, while others may be conflicting in nature.

The bicriteria minimum cost flow problem differs from the classical minimum cost flow problem only in the expression of their objective functions. The bicriteria minimum cost flow problem is shown as below:

Minimize

$$\left\{ \begin{array}{l} \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{(i,j) \in A} d_{ij} x_{ij} \end{array} \right\}$$

subject to

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

Generally, it is observed that these two objectives are conflicting in nature and hence there is no optimal solution in the normal sense for the above problem. The decision maker has to choose solutions, possibly not the best for both the criteria. A special set of solutions, the non-dominated or efficient solutions can be defined to overcome this problem.

An efficient solution is one in which one objective cannot be reduced without a simultaneous increment in the other objective. That is, X^* is an efficient solution to the bicriterion minimum cost flow problem if there does not exist any $X^0 \in S$, the set of all possible solutions such that

$$\begin{aligned} z_1(x^0) &\leq z_1(x^*) \quad \text{and} \\ z_2(x^0) &\leq z_2(x^*) \end{aligned}$$

with atleast one strict inequality.

A number of techniques as mentioned below are available for generating efficient or non-inferior solutions for the bicriteria problem formulated above.

- (i) The weighting method
- (ii) The constraint method
- (iii) The non-inferior set estimation method, and
- (iv) The multi-objective simplex algorithm.

These techniques are discussed in detail by Ambrose Goicoechea [3].

There is one disadvantage with non-dominated solutions technique. It becomes difficult for the decision maker to make his final choice from a set of non-dominated solutions. Another technique known as goal programming, allows the decision maker to specify a target for each of the objective functions. It obtains a preferred solution which is defined as the one that minimizes the sum of the deviations from the prescribed set of target values. A brief description of the goal programming is presented in Sec. 2.5.

An Overview of Constrained Minimum Cost Flow Problem

We now briefly outline the parametric algorithm for the constrained minimum cost flow [CMCF] problem which is the basis for the algorithms developed in this thesis.

The mathematical formulation of the CMCF problem is

$$\text{Min. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (2.1)$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A \quad (2.3)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \leq D \quad (2.4)$$

This algorithm is developed by Ahuja, Batra and Gupta [1]. The algorithm utilizes the concepts of parametric linear programming

and performs all the computations over the network. It first obtains the optimum basis corresponding to C_{\min} , where C_{\min} is the minimum value of C for which a feasible flow exists and then moves from one optimum basis to the next as D is decreased. The algorithm thus yields the minimum cost as a function of the budget, which is a piecewise linear convex function.

Optimality Conditions:

It is well known that the optimum basis of MCF problem is a spanning tree [5]. The addition of the budget constraint introduces one more arc in the optimum basis. It follows from bounded variable linear programming that the necessary and sufficient conditions for a feasible basis structure to be an optimum basis structure are that there exist dual variables π_j 's and $\mu \geq 0$, satisfying the following conditions:

$$(i) \quad \pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in B \quad (2.5)$$

$$(ii) \quad \pi_j - \pi_i \leq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in L \quad (2.6)$$

$$(iii) \quad \pi_j - \pi_i \geq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in U \quad (2.7)$$

$$(iv) \quad \mu (D - \sum_{(i,j) \in A} d_{ij} x_{ij}) = 0 \quad (2.8)$$

where B , L , and U represent the sets of arcs corresponding to the basic variables, the nonbasic variables at their lower bounds, and the nonbasic variables at their upper bounds,

respectively. The set B is called a basis and the triple (B, L, U) is called a basis structure.

Let $(T \cup \{(p, q)\}, L, U)$ be an optimum basis structure at $D = D'$. Further let π_j^c and π_j^d be the numbers satisfying

$$\pi_1^c = 0 \quad \text{and} \quad \pi_j^c - \pi_i^c = c_{ij}, \quad \forall (i, j) \in T \quad (2.9)$$

$$\pi_1^d = 0 \quad \text{and} \quad \pi_j^d - \pi_i^d = d_{ij}, \quad \forall (i, j) \in T \quad (2.10)$$

Define the numbers \bar{c}_{ij} and \bar{d}_{ij} for all arcs $(i, j) \in L \cup U$ as follows:

$$\bar{c}_{ij} = \begin{cases} \pi_i^c - \pi_j^c + c_{ij}, & \text{if } (i, j) \in L \\ \pi_j^c - \pi_i^c - c_{ij}, & \text{if } (i, j) \in U \end{cases} \quad (2.11)$$

$$\bar{d}_{ij} = \begin{cases} \pi_i^d - \pi_j^d + d_{ij}, & \text{if } (i, j) \in L \\ \pi_j^d - \pi_i^d - d_{ij}, & \text{if } (i, j) \in U \end{cases} \quad (2.12)$$

Clearly, $\bar{c}_{ij} = \bar{d}_{ij} = 0$, for all $(i, j) \in T$. If μ is a real number and π_j be the numbers satisfying,

$$\pi_1 = 0 \quad \text{and} \quad \pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall (i, j) \in T$$

then,

$$\pi_j = \pi_j^c + \mu \pi_j^d, \quad \forall j \in N \quad (2.13)$$

substituting (2.13) in the conditions (2.5) - (2.8) and then using (2.11) and (2.12), we obtain the following equivalent conditions:

$$(i) \quad \bar{c}_{pq} + \lambda \bar{d}_{pq} = 0 \quad (2.14)$$

$$(ii) \quad \bar{c}_{ij} + \lambda \bar{d}_{ij} \geq 0 \quad \forall (i,j) \in L \cup U \quad (2.15)$$

$$(iii) \quad \mu (D' - \sum_{(i,j) \in A} \bar{d}_{ij} x_{ij}) = 0 \quad (2.16)$$

The conditions (2.14) - (2.16) are subsequently referred to as the optimality conditions. The \bar{c}_{ij} , \bar{d}_{ij} and μ are referred to as the cost price, budget price and price ratio, respectively, associated with the basis structure $(T \cup \{(p,q)\}, L, U)$.

Depending upon the values of \bar{c}_{ij} and \bar{d}_{ij} , arcs in $L \cup U$ can be classified as follows:

- (i) active arcs: $\{(i,j) \in L \cup U: \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} < 0\}$.
- (ii) critical arcs: $\{(i,j) \in L \cup U: \bar{c}_{ij} \leq 0 \text{ and } \bar{d}_{ij} < 0\}$.
- (iii) passive arcs: $\{(i,j) \in L \cup U: \bar{d}_{ij} \geq 0\}$.

The piecewise linear convex curve between cost and budget is obtained as follows.

Firstly an arc (p, q) belonging to the critical set is selected and added to basis. This results in the formation of exactly one cycle. Next the flow is augmented in this cycle. One of the arcs in the cycles reaches one of its bounds and leaves the basis respectively. The dual variables as well as (B, L, U) are updated and the step is repeated until the critical set is empty. We now describe how the algorithm moves from one basis to the next basis satisfying

the optimality conditions while decreasing the total budget D .

Addition of an arc $(i, j) \in L \cup U$ to the basic tree creates exactly one cycle W_{ij} consisting of the basic arcs. We define the orientation of the cycle W_{ij} along (i, j) if $(i, j) \in L$ and opposite to (i, j) if $(i, j) \in U$. Let W_{ij} and W_{ij}^* be the sets of arcs in the cycle W_{ij} along and opposite to its orientation, respectively. Then using (2.9) and (2.10) it can be easily shown that

$$\tilde{c}_{ij} = \sum_{(i, j) \in W_{ij}} c_{ij} - \sum_{(i, j) \in W_{ij}^*} c_{ij} \quad (2.17)$$

$$\tilde{d}_{ij} = \sum_{(i, j) \in W_{ij}} d_{ij} - \sum_{(i, j) \in W_{ij}^*} d_{ij} \quad (2.18)$$

Thus, \tilde{c}_{ij} (or \tilde{d}_{ij}) denotes the increase in the cost (or budget) if unit amount of additional flow is circulated in the cycle W_{ij} along its orientation. The above given classification of non-tree arcs can be given the physical interpretation. Active arcs are those arcs which can lead to increase in cost if budget is decreased. Critical arcs are those which do not increase cost if budget is reduced. Passive arcs do not lead to decrease in budget even if more cost occurs.

Characteristic Interval:

Let at $D = D_0$, the optimum basis structure of the CMCF problem is $(T \cup \{(p, q)\}, L, U)$. Let \underline{x}_{ij} be the flow on arc $(i, j) \in A$ and \underline{Z} be the cost of this flow. Further, let

c_{ij} , \bar{d}_{ij} and \bar{c}_{ij} be the cost prices, budget prices and price ratio associated with the given basis structure.

We now determine the interval (\underline{D}, \bar{D}) for the values of D for which the given basis structure continues to remain optimum. This interval is known as the characteristic interval associated with $(T \cup \{(p, q)\}, L, U)$.

Since the numbers, \bar{c}_{ij} , \bar{d}_{ij} and \bar{c}_{pq} are uniquely determined for a given basis structure, the optimality conditions (2.14) and (2.15) are not affected by decrease in the value of D . However, since $\lambda \lambda > 0$, the flow must be changed in order to satisfy (2.16). The only way to change the flow, without changing the basis structure and satisfying the flow conservation constraints (2.2), is by circulating some additional flow in the cycle W_{pq} along its orientation. It was noted that $\bar{d}_{pq} < 0$ is the rate at which the budget is reduced and $\bar{c}_{pq} > 0$ is the rate at which the additional cost is incurred when unit amount of flow is circulated. Since the changed flow must also satisfy the bound restrictions of the arcs (2.3), we calculate the maximum flow \bar{f} that can be circulated without violating the bound restrictions of the arcs in W_{pq} . If we define,

$$\bar{f}_{ij} = \begin{cases} b_{ij} - x_{ij} , & \text{if } (i, j) \in \bar{W}_{pq} \\ x_{ij} , & \text{if } (i, j) \in W_{pq} \end{cases}$$

then,

$$\bar{f} = \min_{(i,j) \in W_{pq}} \{f_{ij}\}$$

Thus,

$$\bar{D} = \underline{D} + \bar{f} \bar{d}_{pq}$$

For all values of D in (\underline{D}, \bar{D}) , the optimum flow is given by,

$$x_{ij} = \begin{cases} \bar{x}_{ij} + \alpha' \bar{f}, & \text{if } (i,j) \in W_{pq} \\ \underline{x}_{ij} - \alpha' \bar{f}, & \text{if } (i,j) \in W_{pq} \\ x_{ij}, & \text{otherwise} \end{cases}$$

and cost by

$$Z = \underline{Z} + \alpha' \bar{f} \bar{c}_{pq},$$

$$\text{where, } \alpha' = (\bar{D} - \underline{D}) / (\bar{D} - \underline{D})$$

Let \bar{x}_{ij} denote the flow in arc $(i,j) \in A$ at $D = \bar{D}$. If it is required to find the optimum flow for $D < \bar{D}$, then a dual simplex iteration is performed to obtain a new basis structure at $D = \bar{D}$.

The Dual Simplex Iteration:

At $D = \bar{D}$, flow in an arc $(u, v) \in W_{pq}$, for which $\bar{f}_{uv} = \bar{f}$, equals its lower or upper bound. If D is decreased further, the bound is violated. Thus, to obtain a new basis structure at $D = \bar{D}$, which may permit decrease in the value of D , the arc (u, v) is dropped from the basis and a non-basic arc

is selected to enter the basis. $T \cap \text{arc } (v, v)$ becomes a non-basic arc at its respective bound. In the new basis structure, the dashed values and sets represent the corresponding values and sets of the previous basis structure. By selecting an active arc and performing dual simplex iteration each time until the active set is empty, the cost is increased and the budget is reduced.

The curve starts at C_{\min} and as the value of D is decreased, the value of Z keeps increasing until a value of D is reached when Z stops increasing. The slope of the curve at any point is the value of $-\lambda$ in the optimum solution which corresponds to that point. Since the value of $-\lambda$ keeps increasing and finally becomes zero and no critical arcs are formed at any iteration (The proof is given in [1]), the curve is a piecewise linear convex function.

2.5 An Overview of Goal Programming:

As mentioned in the previous section, Goal programming allows the decision maker to specify a target for each objective function which provides him the preferred solution by minimizing the sum of the deviations from the prescribed set of target values.

Generalized goal programming has a number of special terms and concepts that are being used in this thesis. They are mentioned as below:

(i) Objective:

An objective is a relatively general statement (in narrative or quantitative terms) that reflects the desires of the decision maker. For example, one may wish to "maximize profit" or "minimize total wastage" or "wipe out poverty".

(ii) Aspiration Level:

An aspiration level is a specific value associated with the desired or acceptable level of achievement of an objective. Thus, an aspiration level is used to measure the achievement of an objective and generally serves to "anchor" the objective to reality.

(iii) Goal:

An objective in conjunction with an aspiration level is termed a goal. For example, we may wish to "achieve at least X units of profits" or "reduce the rate of inflation by Y percent."

(iv) Goal Deviations:

The difference between what one accomplishes and what one aspires to is the deviation from his goal. A deviation can represent over as well as under achievement of a goal.

(v) Goal Formulation:

We will now examine how to mathematically transform an objective into a goal within our goal programming frame work.

Consider the objective function expressed in general terms as $f_i(X)$. The procedure to be presented is applicable whether $f_i(X)$ is linear or nonlinear, but only linear objectives are considered in this thesis.

We then let,

$f_i(X)$ = mathematical representation of objective i
as a function of the decision variables

$$X = (x_1, x_2, \dots, x_n)$$

b_i = value of the aspiration level associated with objective i .

Three possible forms of goals may then result:

- (i) $f_i(X) \leq b_i$: that is, we wish to have a value of $f_i(X)$ that is equal to or less than b_i .
- (ii) $f_i(X) \geq b_i$: that is, we wish to have a value of $f_i(X)$ that is equal to or greater than b_i .
- (iii) $f_i(X) = b_i$: that is $f_i(X)$ must exactly equal b_i .

Regardless of the form, we shall transform any of these relations into the goal programming format by adding

a negative deviation variable ($\beta_j \geq 0$) and subtracting a positive deviation variable ($\alpha_j \geq 0$). This statement is summarized as below:

Goal Type	Goal Programming Form	Deviation Variables to be Minimized
$f_i(x) \leq b_i$	$f_i(x) + \beta_i - \alpha_i = b_i$	α_i
$f_i(x) \geq b_i$	$f_i(x) + \beta_i - \alpha_i = b_i$	β_i
$f_i(x) = b_i$	$f_i(x) + \beta_i - \alpha_i = b_i$	$\alpha_i + \beta_i$

We will briefly discuss the three extensions of goal programming, They are:

- (1) Weighted goal programming.
- (2) Interval goal programming.
- (3) Fuzzy goal programming.

Weighted Goal Programming:

In this model the decision maker assigns an aspiration level for each of the objectives and also the weighting factors for each of the deviations. It obtains an optimum solution by minimizing the sum of weighted deviations.

There is an alternative way in which the weighted model may be formed. Rather than multiplying each deviation variable by a constant weight, we may instead, raise each

deviation variable in the achievement function to some power. This results in a polynomial form for the achievement function.

Given a multi-objective model,

optimize Z_i $i = 1, \dots, s$

s.t.

$$AX \leq b$$

$$X \geq 0$$

Adding aspiration levels and deviations variables to each objective and weighting each resultant goal, we obtain,

$$\text{Minimize } a = \sum_{i=1}^s (w_i \alpha_i + r_i \beta_i)$$

subject to

$$Z_i(X) + \beta_i - \alpha_i = Z_i^0, \quad i = 1, \dots, s$$

$$AX \leq b$$

$$X, \alpha_i, \beta_i \geq 0, \quad i = 1, \dots, s$$

where,

α_i = Weighting factor for the positive deviation of goal i .

β_i = Weighting factor for the negative deviation of goal i .

Z_i^0 = Aspiration level for objective i .

Interval Goal Programming:

In this model the decision maker specifies a range of aspiration levels for each of the goals instead of one

aspiration level. It obtains an optimum solution by minimizing the weighted sum of deviations from the set of ranges.

The mathematical formulation of this model is,

$$\text{Minimize } a = \sum_{i=1}^s (w_{i,2} \alpha_{i,2} + r_{i,1} \beta_{i,1})$$

subject to

$$z_{i,1} + \beta_{i,1} - \alpha_{i,1} = L_i \quad \text{all } s$$

$$z_{i,2} + \beta_{i,2} - \alpha_{i,2} = U_i \quad \text{all } s$$

where,

$z_{i,1} = z_{i,2}$ = expression for objective k.

$\beta_{i,1}, \beta_{i,2}$ = negative deviations.

$\alpha_{i,1}, \alpha_{i,2}$ = positive deviations.

$r_{i,1}$ = weighting factor for the negative deviation for goal $z_{i,1}$.

$w_{i,2}$ = weighting factor for the positive deviation for goal $z_{i,2}$.

Fuzzy Linear Programming.

A fairly recent attempt at modeling and solving the multiple-objective problem is that known as fuzzy programming. The approach is similar, in many respects, to the weighted linear goal programming method previously discussed, differing primarily in the manner in which the importance of the goals

are considered. This method minimizes the worst under achievement of any goal.

A major advantage of fuzzy linear programming is that it may be transformed into a conventional linear programming model. The main disadvantage of this method is, the underachievement of just one goal can have a major impact on the solution, since it attempts to minimize the maximum underachievement.

CHAPTER III

NETWORK GOAL PROGRAMMING ALGORITHMS

3.1 Introduction:

In this chapter, we shall consider two special classes of linear goal programming problems, i.e. weighted goal network flow problem and interval goal network flow problem.

We generally confront problems that require two objectives to be considered simultaneously. Often, these objectives are conflicting in nature. These types of problems may arise in road networks, communication networks and pipe line networks. The typical applications of bi-criteria network flow problems are mentioned in Section 1.1.

One of the methods of solving bi-criteria problems is to obtain the efficient or non-dominated solutions set. As the efficient solution set is considerably large, it becomes practically difficult for the decision maker to choose the solution he would prefer.

Goal programming is a technique which takes the decision maker's preference into consideration and provides him the preferred solution. He expresses his preference by specifying targets to the objectives and the weights to the deviations from the targets.

In weighted goal programming, the decision maker specifies a target for each of the objective and weights for the deviations. It provides him an optimum solution which is as close as possible to the specified targets, by minimizing the sum of weighted deviations.

In interval goal programming the decision maker specifies a range or interval for each of the objectives and weights for each of the deviations from lower and upper bounds of the interval. It obtains an optimum solution lying in the ranges specified if one exists or finds a solution such that the sum of weighted deviations are minimum.

This chapter has been divided into 11 sections. In Sec. 2 we present the notations used to represent the feasible region of the bi-criteria network flow problem. The tracing of various trade-off curves is presented in Sec. 3. The mathematical formulation and development of the WGNF algorithm are presented in Sec. 4 and 5. In Sec. 6 and Sec. 7 we present the statement of the WGNF algorithm and a numerical example. The mathematical formulation and development of IGFN algorithm are given in Sec. 8 and Sec. 9. A numerical example of IGFN problem is given in Sec. 10. Finally the computational performance of both the algorithms is reported in Sec. 11.

3.2 Notations:

The various regions as shown in Fig. 3.1 are defined as below:

$$E : \{(x, y) : x \geq 0 \text{ and } y \geq 0\}.$$

$$S : \{(x, y) : x = \sum_{(i, j) \in A} c_{ij} x_{ij}, \quad y = \sum_{(i, j) \in A} d_{ij} x_{ij}\}.$$

$$R_1 : \{(x, y) : 0 \leq x \leq f_1, \quad 0 \leq y \leq g_2 \text{ and } (x, y) \notin S\}$$

$$R_2 : \{(x, y) : 0 \leq x \leq f_3, \quad g_2 \leq y \leq \infty \text{ and } (x, y) \notin S\}$$

$$R_3 : \{(x, y) : f_3 \leq x \leq \infty, \quad g_4 \leq y \leq \alpha \text{ and } (x, y) \notin S\}$$

$$R_4 : \{(x, y) : f_1 \leq x \leq \infty, \quad 0 \leq y \leq g_4 \text{ and } (x, y) \notin S\}$$

$$S_1 : \{(x, y) \in S : x \leq f_1\}$$

$$S_2 : \{(x, y) \in S : x \leq f_3\}$$

$$S_3 : \{(x, y) \in S : x > f_3\}$$

$$S_4 : \{(x, y) \in S : x > f_1\}$$

α_i : positive deviation from goal i.

β_i : negative deviation from goal i.

w_i : weight assigned to positive deviation α_i .

r_i : weight assigned to negative deviation β_i .

$w_i, r_i > 0$.

3.3 Tracing the Trade-Off Curves:

The optimum solutions for WGNF problem and IGNF problem lie either on B_1, B_2, B_3 and B_4 curves or in the feasible region S as shown in Fig. 3.1. Since our algorithm searches for an optimum solution by traversing along $B_1 \cup B_2 \cup B_3 \cup B_4$

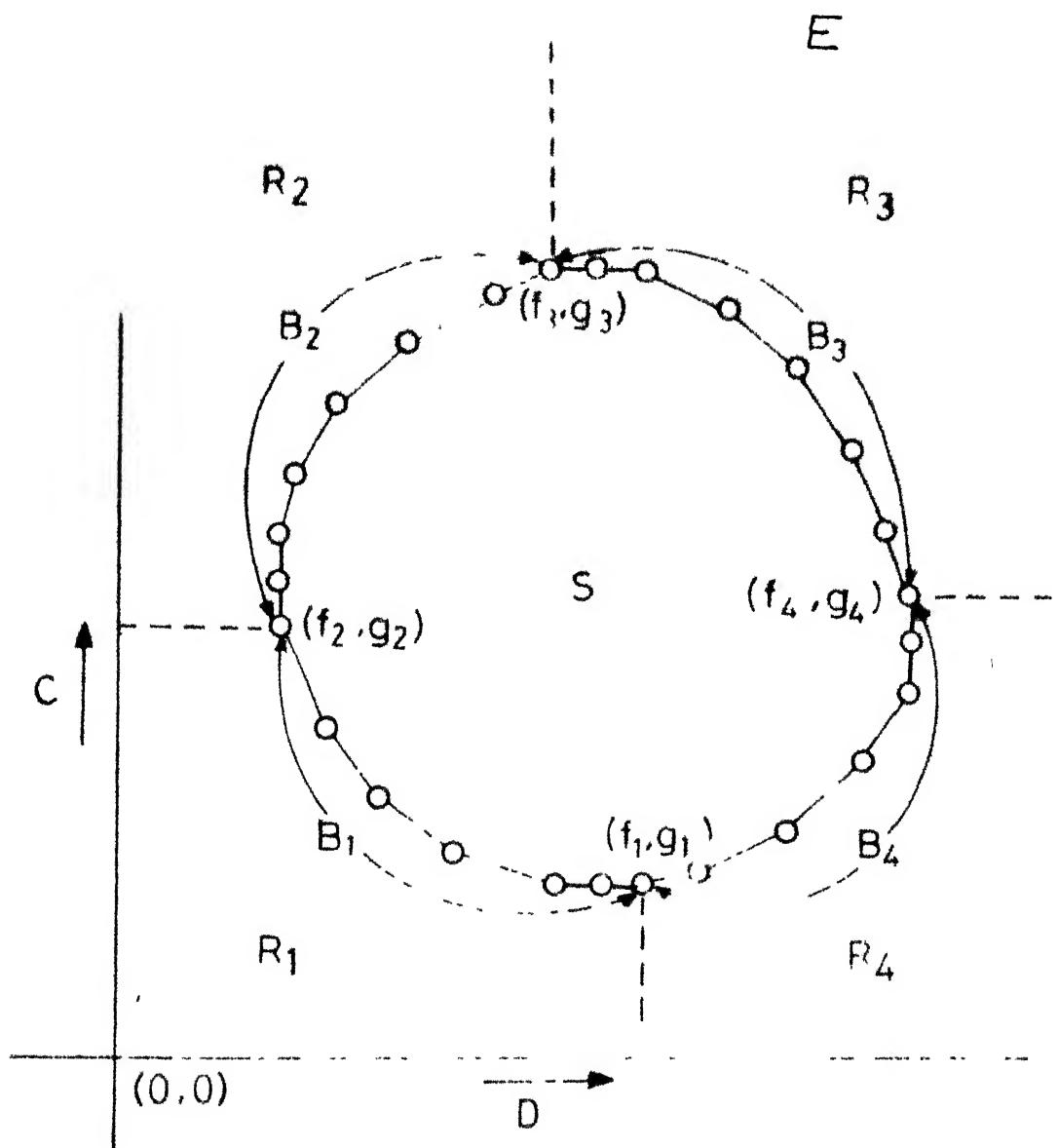


Fig. 3.1 Feasible region of bicriteria network flow problem

and through the feasible region S it is necessary to trace the B_1 , B_2 , B_3 and B_4 curves. Once these curves are obtained, it can be shown that they together form a closed convex feasible region S .

Tracing B_1 Curve:

The detailed procedure for obtaining the B_1 curve was presented in Sec. 2.4. The procedure for tracing B_2 , B_3 and B_4 curves is same as that for tracing B_1 , except that the optimality conditions, the critical set, the active set, the passive set and the criteria for selecting the price ratio differ, as the problems considered to obtain each of these curves are different.

Tracing B_2 Curve:

B_2 curve is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the following problem.

$$\text{Max. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

subject to

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \leq D$$

The optimality conditions are:

$$(i) \quad \bar{c}_{pq} - \mu \bar{d}_{pq} = 0$$

$$(ii) \quad -\bar{c}_{ij} + \mu \bar{d}_{ij} > 0 \quad \forall (i,j) \in L \cup U$$

$$(iii) \quad \mu(D - \sum_{(i,j) \in A} d_{ij} x_{ij}) = 0$$

The non-basic arcs are classified as:

$$(i) \quad \text{Active arcs } S' : \{(i,j) \in L \cup U : \bar{c}_{ij} < 0 \text{ and } \bar{d}_{ij} < 0\}$$

$$(ii) \quad \text{Critical arcs } S'' : \{(i,j) \in L \cup U : \bar{c}_{ij} \geq 0 \text{ and } \bar{d}_{ij} < 0\}$$

$$(iii) \quad \text{Passive arcs } S''' : \{(i,j) \in L \cup U : \bar{d}_{ij} \geq 0\}$$

Initially, $\sum_{(i,j) \in A} c_{ij} x_{ij}$ is maximized and let (C_{\max}, D) be the solution. A critical arc $(p,q) \in S''$ is selected and entered into the basis and dual-simplex iteration is performed. This is continued until S is empty.

Next, an active arc (p,q) that $\mu_{pq} = \min_{(i,j) \in S'} (\mu_{ij})$ is selected and entered into the basis and dual-simplex iteration is performed. Each dual-simplex iteration results in an extreme solution. This is repeated until $S' = \{\emptyset\}$. As μ is gradually increasing and ultimately becomes zero along with gradual decrease in C and D , the curve B_2 is a piecewise concave linear function.

Tracing B_3 Curve.

B_3 is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the

following problem,

$$\text{max. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

subject to

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \geq D$$

The optimality conditions are:

$$(i) \quad \bar{c}_{pq} + \mu \bar{d}_{pq} = 0$$

$$(ii) \quad -\bar{c}_{ij} - \mu \bar{d}_{ij} \geq 0 \quad \forall (i,j) \in A$$

$$(iii) \quad \mu \left(\sum_{(i,j) \in A} d_{ij} x_{ij} - D \right) = 0$$

The non basic-arcs are classified as:

(i) Active arcs $S' : \{(i,j) \in L \cup U : \bar{c}_{ij} < 0 \text{ and } \bar{d}_{ij} > 0\}$

(ii) Critical arcs $S'' : \{(i,j) \in L \cup U : \bar{c}_{ij} \geq 0 \text{ and } \bar{d}_{ij} > 0\}$

(iii) Passive arcs $S''' : \{(i,j) \in L \cup U : \bar{d}_{ij} \leq 0\}$

Initially, $\sum_{(i,j) \in A} c_{ij} x_{ij}$ is maximized and let (c_{\max}, D) be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc (p,q) such that

$\mu_{pq} = \min_{(i,j) \in S} (\lambda_{ij})$ is selected and entered into the basis.

The dual simplex iteration is performed. This is repeated until

$S' = \{\emptyset\}$. Since the value of μ is gradually increasing and C is gradually decreasing with the increase in D , the curve B_3 is a piecewise concave function.

Tracing B_4 Curve:

B_4 curve is a piecewise linear convex function between cost and budget. This curve is obtained by parametrically solving the following problem:

$$\text{Min. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \geq D$$

The optimality conditions are.

$$(i) \quad \bar{c}_{pq} - \mu \bar{d}_{pq} = 0$$

$$(ii) \quad \bar{c}_{ij} - \mu \bar{d}_{ij} > 0 \quad \forall (i,j) \in L \cup U$$

$$(iii) \quad \mu \left(\sum_{(i,j) \in A} d_{ij} x_{ij} - D \right) = 0$$

The non-basic arcs are classified as:

(i) Active arcs $S' : \{(i,j) \in L \cup U : \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} > 0\}$

(ii) Critical arcs $S'' : \{(i,j) \in L \cup U : \bar{c}_{ij} \leq 0 \text{ and } \bar{d}_{ij} > 0\}$

(iii) Passive arcs S'' : $\{(i,j) \in L \cup U : \bar{d}_{ij} \leq 0\}$

Initially $\sum_{(i,j) \in A} c_{ij} x_{ij}$ is minimized and let (C_{\min}, D) be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc (p,q) such that $u_{pq} = \min_{(i,j) \in S} (u_{ij})$ is selected and entered into the basis. The dual simplex iteration is performed. This is repeated until $S' = \{\emptyset\}$. A dual simplex iteration results in an extreme solution. Since the value of u is gradually increasing and both C and D are increasing, the curve B_4 is a piecewise convex function.

The curves B_1 and B_4 together constitute a convex curve as the starting solution is same and the cost is gradually increasing. The curves B_2 and B_3 together constitute a concave curve as the starting solution is the same and the cost is gradually decreasing. It follows from the above discussion that the terminating solution of B_1 is the starting solution for B_2 and the terminating solution of B_2 is the starting solution for B_3 . Similarly the terminating solution of B_3 is the starting solution for B_4 and the terminating solution of B_4 is the starting solution for B_1 . Thus we obtain a closed region enclosed by the curves $B_1 \cup B_2 \cup B_3 \cup B_4$. Since $B_1 \cup B_4$ is a convex curve and $B_2 \cup B_3$ is a concave curve, the closed feasible region is a closed convex region.

3.4 Mathematical Formulation of the Weighted Goal Network Flow Problem:

In this section, we give the mathematical formulation of the weighted goal network flow problem.

In the network $G = (N, A)$, associated with each arc $(i, j) \in A$ are two numbers c_{ij} and d_{ij} . The capacity of each arc $(i, j) \in A$ is b_{ij} . The amount that is to be shipped from source l to sink n is V .

$$\text{Minimize } Z = w_1\alpha_1 + r_1\beta_1 + w_2\alpha_2 + r_2\beta_2 \quad (3.1)$$

subject to the following constraints:

Flow Conservation Constraints:

These constraints essentially represent the fact that flow of the commodity is conserved at all nodes, except at source and sink.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=l \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (3.2)$$

Capacity Constraints:

These constraints essentially represent the fact that flow over an arc (i, j) cannot exceed its capacity

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i, j) \in A \quad (3.3)$$

Goal Constraints:

An objective in conjunction with an aspiration level and deviations is known as goal constraint.

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1 \quad (3.4)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1 \quad (3.5)$$

3.5 Development of the Algorithm:

We shall now discuss, the various steps involved in finding an optimum solution for a specified $(C_1, D_1, w_1, r_1, w_2, r_2)$. Let us consider $(C_1, D_1) \in R_2$ in order to trace all the possible steps of the WGNF algorithm.

Initially, the minimum cost flow problem with $\sum_{(i,j) \in A} c_{ij} x_{ij}$ as an objective function is solved. Let (C_{\min}, D') be the minimum cost and budget obtained and it is represented by I in Fig. 3.3. The deviations axes from aspiration levels (C_1, D_1) are shown in Fig. 3.2. If the algorithm traverses in quadrant I then it minimizes α_1 and α_2 and the other two deviations are zero. Similarly α_1 and β_2 are minimized in quadrant II, β_2 and β_1 are minimized in quadrant III and α_2 and β_1 are minimized in quadrant IV. Therefore, the algorithm always minimizes only two deviations and the other two being at zero. The above statement is supported by theorems 1 to 5 given below.

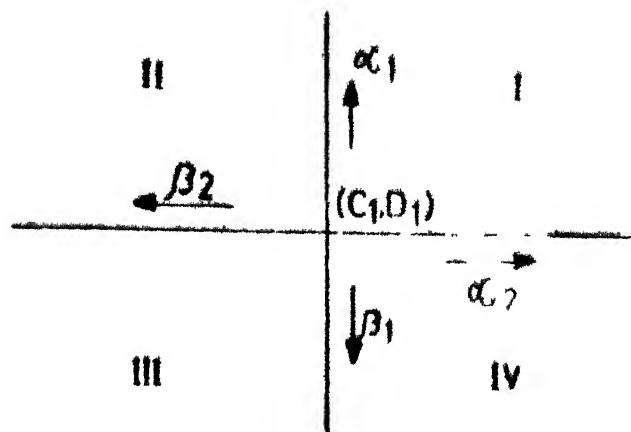


Fig. 3.2 Aspiration levels and deviations for WGNF problem

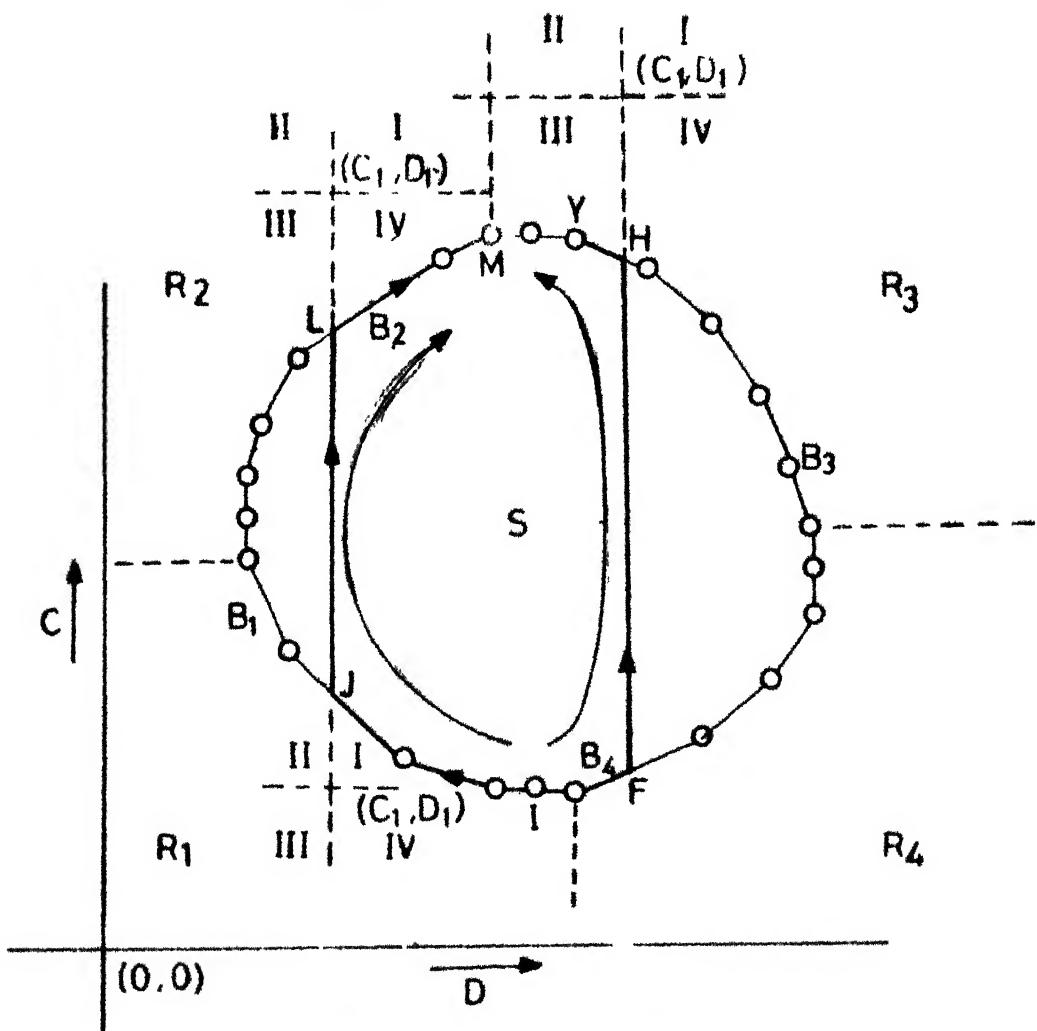


Fig. 3.3 Paths traced by WGNF regorithm to obtain optimum solutions

THEOREM 1: $\alpha_i \beta_i = 0, \forall i = 1, 2$

Proof: We will prove this theorem by contradiction.

Suppose $(x^*, \alpha_1^*, \beta_1^*)$ be an optimum solution and $\alpha_1^*, \beta_1^* > 0$.

Let $C^* = \sum_{(i,j) \in A} c_{ij} x_{ij}^*$

The value of the objective function is

$$Z_1 = w_1 \alpha_1^* + r_1 \beta_1^* \quad (3.6)$$

The goal constraint is

$$C^* + \beta_1 - \alpha_1 = C_1 \quad (3.7)$$

Let $\Delta = \min. (\beta_1^*, \alpha_1^*)$

Consider the solution $(x^*, \beta_1', \alpha_1')$,

$$\text{let, } \beta_1' = \beta_1^* - \Delta \quad (3.8)$$

$$\alpha_1' = \alpha_1^* - \Delta \quad (3.9)$$

substituting (3.8) and (3.9) in (3.6) and (3.7), we get,

$$\begin{aligned} Z_2 &= w_1 (\alpha_1' + \Delta) + r_1 (\beta_1' + \Delta) \\ &= w_1 \alpha_1' + r_1 \beta_1' + \Delta (w_1 + r_1) \end{aligned} \quad (3.10)$$

goal constraint is

$$C^* + \beta_1' - \alpha_1' = C_1 \quad (3.11)$$

Hence the goal constraint is satisfied.

Since $\Delta, w_1, r_1 > 0$ it follows that $Z_2 > Z_1$. This contradicts that $(x^*, \beta_1^*, \alpha_1^*)$ is an optimum solution. Therefore $\Delta = 0$. Hence the theorem follows.

Similarly it can be proved for $\alpha_2 \beta_2 = 0$. Therefore atleast two of $\alpha_1, \beta_1, \alpha_2$ and β_2 are zero.

QED

THEOREM 2: If $(C_1, D_1) \in R_1$, then $(C^*, D^*) \in B_1$ and $D^* \geq D_1$ and $C^* \geq C_1$.

Proof: Consider $(C', D') \in S_2$. There exists $(C, D) \in S_1$, such that $D < D'$. Since $D < D'$, it follows that $\alpha_2 < \alpha'_2$ resulting in $Z < Z'$ where Z is the value of objective function (C, D) . Hence a better solution exists in S_1 for every solution in S_2 .

Consider a feasible solution $(C', D') \in S_1$. A feasible solution $(C, D') \in B_1$ exists such that $C < C'$. Since $C < C'$ it follows that $\alpha_1 < \alpha'_1$. It results in $Z < Z'$ where Z and Z' are objective functions of (C, D') and (C', D') respectively. Therefore for any solution in S , a better solution exists on B_1 .

Let $(C', D') \in B_1$ and $D' < D_1$. There exists $(C, D_1) \in B_1$ and $C < C'$. Therefore $\alpha_1 < \alpha'_1$ resulting in $Z < Z'$.

Let $(C'', D'') \in B_1$ and $C'' < C_1$. There exists (C_1, D) and $C_1 > C''$. Therefore $\alpha_2 < \alpha''_2$ and $\beta_1 < \beta''_1$, resulting $Z < Z''$. Hence an optimum solution is $(C^*, D^*) \in B_1$ and $D^* \geq D$, and $C^* \geq C_1$. It follows from theorem 1 and above that if $(C_1, D_1) \in R_1$ then the optimum solution is $(X^*, \alpha_1^*, \alpha_2^*)$ and $\beta_1, \beta_2 = 0$.

Similarly we can prove the following theorems.

QED

THEOREM 3: If $(C_1, D_1) \in R_2$ then $(C^*, D^*) \in B_2$ and $D^* \geq D_1$ and $C^* \leq C_1$.

The optimum solution is $(X^*, \alpha_2^*, \beta_1^*)$ and $\alpha_1 = 0$.

THEOREM 4: If $(C_1, D_1) \in R_3$, then $(C^*, D^*) \in B_3$ and $D^* \leq D_1$ and $C^* \leq C_1$.

The optimum solution is $(X^*, \beta_1^*, \beta_2^*)$ and $\alpha_1 = 0$.

THEOREM 5: If $(C_1, D_1) \in R_4$, then $(C^*, D^*) \in B_4$ and $D^* \leq D_1$ and $C^* \geq C_1$.

The optimum solution is $(X^*, \beta_2^*, \alpha_1^*)$ and $\alpha_2 = 0$.

QED

If $D_1 \leq D'$ then the algorithm moves along the B_1 curve in the direction of J as shown in Fig. 3.3. Otherwise it traverses along B_4 . The algorithm evaluates the objective function values of extreme-solutions and two non-extreme solutions, namely $(C, D_1) \in B_1 \cup B_2 \cup B_3 \cup B_4$ and $(C_1, D) \in B_1 \cup B_2 \cup B_3 \cup B_4$, because one of these solutions is optimum if $(C_1, D_1) \notin S$. We ascertain the above statement in theorems 6 to 9 given below.

THEOREM 6: If $(C_1, D_1) \notin S$, then one of the extreme solutions of $B_1 \cup B_2 \cup B_3 \cup B_4$ is optimum.

Proof: Consider two adjacent extreme solutions (C', D') and (C'', D'') . Any non-extreme solution (C, D) lying between (C', D') and (C'', D'') can be expressed as convex combination of

(C', D') and (C'', D'') . Therefore Z' or Z'' is atleast as good as any \bar{Z} , depending on $Z' \leq Z''$ or $Z'' \leq Z'$. Hence the theorem follows.

QED

THEOREM 7: If $(C_1, D) \in B_1$ lies between two adjacent extreme solutions $(C', D') \in B_1$ and $(C'', D'') \in B_1$ such that $C_1 > C'', D' > D_1$ and satisfying $Z' \geq Z$ where Z' , Z and Z'' are objective function values of (C', D') , (C_1, D) and (C'', D'') respectively, then (C_1, D) is optimum.

Proof: $Z'' = w_2 \alpha_2'' + r_1 \beta_1'' \quad (3.12)$

$$Z = w_2 \alpha_2$$

Since $\alpha_2 < \alpha_2''$, it follows that $Z < Z''$. (C_1, D) is atleast as good as any other solution lying between (C_1, D) and (C', D') as they can be expressed as a convex combination of (C_1, D) and (C', D') and $Z' \geq Z$. Similarly (C_1, D) is better than any other solution lying between (C_1, D) and (C'', D'') because they can be expressed as a convex combination of (C_1, D) and (C'', D'') and $Z'' > Z$. According to theorem 2, for any solution in S , a better solution exists on B_1 . Hence (C_1, D) is atleast as good as any neighbouring solution. Therefore (C_1, D) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of $(C_1, D) \in B_4$, $(C, D_1) \in B_2$ and $(C, D_1) \in B_3$ lying between two adjacent extreme solutions, (C', D') and (C'', D'') satisfying $Z' \geq Z$ then they are optimum.

QED

THEOREM 8: If $(C, D_1) \in B_1$ lies between two adjacent extreme solutions (C', D') and (C'', D'') in B_1 such that $C > C'', C'' > C_1$ and satisfying $Z < Z''$ where Z, Z', Z'' are objective function values of (C, D_1) , (C', D') and (C'', D'') respectively, then (C, D_1) is optimum.

$$\text{Proof: } Z' = w_1 \alpha'_1 + r_2 \beta'_2$$

$$Z = w_1 \alpha_1$$

Since $\alpha_1 < \alpha'_1$, it follows that $Z < Z'$. (C, D_1) is a better than any solution between (C', D') and (C, D_1) and between (C, D_1) and (C'', D'') as they can be expressed as convex combination of these solutions and $Z < Z'$ and $Z < Z''$. According to theorem 2, for any solution in S , a better solution exists on B_1 . Hence (C, D_1) is atleast as good as any neighbouring solution. Therefore (C, D_1) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of $(C, D_1) \in B_4$, $(C_1, D) \in B_2$ and $(C_1, D) \in B_3$ lying between two

adjacent extreme solutions, (C', D') and (C'', D'') satisfying $Z < Z''$ then they are optimum.

QED

THEOREM 9: If $(C, D_1) \in B_1$ and $(C_1, D) \in B_1$ lie between two adjacent extreme solutions (C', D') $\in B_1$ and $(C'', D'') \in B_1$ such that $C_1 > C''$, then

- (i) (C_1, D) is optimum if $Z \leq Z'$ and
- (ii) (C, D_1) is optimum if $Z \geq Z''$

where Z and Z'' are the objective function values of (C_1, D) and (C, D_1) .

Proof: We shall prove case (i).

Let Z' and Z'' be objective function values of (C', D') and (C'', D'')

$$Z'' = w_2 \alpha_2'' + r_1 \beta_1''$$

$$Z = w_2 \alpha_2$$

Since $\alpha_2 < \alpha_2''$, it follows that $Z < Z''$. (C_1, D) is atleast as good as any solution lying between (C_1, D) and (C, D_1) because they can be expressed as convex combination of (C_1, D) and (C, D_1) and $Z \leq Z''$. Similarly (C_1, D) is better than any solution lying between (C_1, D) and (C'', D'') as they can be expressed as convex combination of (C_1, D) and (C'', D'') and $Z < Z''$. According to theorem 2, for any solution in S , there exists a better solution on B_1 . Hence (C_1, D) is atleast as

good as any neighbouring solution. Therefore (C_1, D) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Case (ii) is proved as follows:

$$Z' = w_1 \alpha'_1 + r_2 \beta'_2$$

$$\bar{Z} = w_1 \bar{\alpha}_1$$

Since $\bar{\alpha}_1 < \alpha'_1$, it follows that $\bar{Z} < Z'$ using theorem 2, $\bar{Z} < Z$ and $\bar{Z} \leq Z$, it can be proved that (C, D_1) is a local optima, which is also a global optima in linear programming. Hence (C, D_1) is an optimum solution.

Similar results can be proved for the cases when $(C, D_1) \in B_2 \cup B_3 \cup B_4$ and $(C_1, D) \in B_2 \cup B_3 \cup B_4$ lie between two adjacent extreme solutions then one of them is optimum depending on $\bar{Z} \geq Z$ or $\bar{Z} < Z$.

QED

We can infer from theorems (7) to (9) that the non-extreme solutions (C, D_1) and (C_1, D) behave as extreme solutions and hence the algorithm evaluates the objective function values of these non-extreme solutions.

If Z corresponding to (C, D) the current solution is greater than or equal to Z'' of (C'', D'') the previous extreme solution, then (C'', D'') is optimum and the algorithm stops. This is proved in the following theorem.

THEOREM 10: If $(C_1, D_1) \in R_1$ and $(C', D') \in B_1$, $(\bar{C}, \bar{D}) \in B_1$ and $(C'', D'') \in B_1$ be three adjacent extreme solutions satisfying $Z'' \geq \bar{Z}$ and $Z' \geq \bar{Z}$ then (\bar{C}, \bar{D}) is an optimum solution.

Proof: (\bar{C}, \bar{D}) is atleast as good as any other solution lying between (\bar{C}, \bar{D}) and (C', D') as they can be expressed as convex combination of these two solutions and $Z' \geq \bar{Z}$. Similarly (\bar{C}, \bar{D}) is atleast as good as any other solution lying between (\bar{C}, \bar{D}) and (C'', D'') because they can be expressed as convex combination of these two solutions and $Z'' \geq \bar{Z}$. According to theorem 2, if $(C_1, D_1) \in R_1$ then a better solution exists on B_1 than any solution in S . Therefore (\bar{C}, \bar{D}) is a local optima, as it is better than any neighbouring solution. In linear programming local optima is also global optima and hence (\bar{C}, \bar{D}) is optimum. Similarly, this type of result can be proved for the cases, $(C_1, D_1) \in R_2 \cup R_3 \cup R_4$.

QED

If $D_1 \leq D_{\min}$, then the algorithm moves along $B_1 \cup B_2$ and obtains an optimum solution. Otherwise the algorithm halts at $(C, D_1) \in B_1$ represented by J in Fig. 3.3. If $(C_1, D_1) \in R_1$ then (C, D_1) is an optimum solution which follows from the theorem 8. Otherwise it moves vertically up along JL in the direction of C_1 .

We will discuss in detail how the algorithm increases the total cost C , while maintaining the total budget D constant. This is achieved by simultaneously augmenting the flow in two cycles.

An arc (i, j) is a forward arc if the flow is sent in the direction of the arc, and a backward arc if the flow is sent against the direction of the arc. We will refer to this as orientation of the arc. The basis structure of current solution is $(T \cup \{(p, q)\}, L, U)$ and its basis consists of exactly one cycle formed by the arc (p, q) . This cycle is denoted by W_{pq} . Now, define a set

$$S^0 = \{(i, j) \in L \cup U : \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and}$$

$$(\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{ij}} \times \bar{c}_{ij} > 0) \}.$$

The above conditions are derived as follows.

Criteria for Selecting a Non-Basic Arc for 2 Cycle Flow Augmentation:

We will derive the conditions that the non-basic arc $(r, s) \in L \cup U$ must satisfy, such that by simultaneously augmenting flow in cycles W_{pq} and W_{rs} , the total cost increases and the total budget remains constant.

The basis structure is $(T \cup \{(p, q)\}, L, U)$.

Let Δ_{pq} be the flow to be circulated in cycle W_{pq} .

Let Δ_{rs} be the flow to be circulated in cycle W_{rs} .

For the total budget D to remain constant,

$$\Delta_{pq} \bar{d}_{pq} + \Delta_{rs} \bar{d}_{rs} = 0 \quad (3.13)$$

In order to increase the total cost C ,

$$\Delta_{pq} \bar{c}_{pq} + \Delta_{rs} \bar{c}_{rs} > 0 \quad (3.14)$$

Using (3.13),

$$\frac{\Delta_{rs}}{\Delta_{pq}} = - \left(\frac{\bar{d}_{pq}}{\bar{d}_{rs}} \right) = K \quad (3.15)$$

Since $\Delta_{pq}, \Delta_{rs} > 0$, hence in order to satisfy (3.13), we get,

$$\frac{\bar{d}_{pq}}{\bar{d}_{rs}} < 0 \quad (3.16)$$

Dividing (3.14) by Δ_{pq} and substituting (3.15), we get,

$$\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{rs}} \times \bar{c}_{rs} > 0 \quad (3.17)$$

Hence if an arc (r, s) satisfies (3.16) and (3.17) and augmenting flow Δ_{pq} and Δ_{rs} in cycles W_{pq} and W_{rs} , the total cost increases and total budget D remains constant.

If $S^0 = \{\phi\}$ then reverse the orientation of the arc (p, q) and set $\bar{c}_{pq} = -\bar{c}_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$. Then, we again find S^0 . If S^0

is not empty, select an arc $(r, s) \in S^0$ and enter it into the basis. This results in the formation of two cycles W_{pq} and W_{rs} . The orientation of a cycle W_{ij} , $(i, j) \in L \cup U$ is same as that of the arc (i, j) . Let \bar{W}_{ij} and \underline{W}_{ij} be the sets of arcs, along and opposite to the orientation of the cycle W_{ij} , respectively. Let Δ_{pq} be the amount of flow to be augmented in W_{pq} and $K \cdot \Delta_{pq}$ be the flow to be augmented in cycle W_{rs} where $K = - \bar{d}_{pq}/\bar{d}_{rs}$. Next, we determine Δ_{pq} as follows.

Consider an arc $(i, j) \in (W_{pq} \cap W_{rs})$. The flow in arc (i, j) after augmenting flow in W_{pq} and W_{rs} is,

$$\begin{aligned}\tilde{x}_{ij} &= x_{ij} + \Delta_{pq} I_1 + I_2 K \Delta_{pq} \\ &= x_{ij} + \Delta_{pq} (I_1 + I_2 K)\end{aligned}$$

where,

$$x_{ij} = \text{existing flow in } (i, j)$$

$$I_1 = +1, \text{ if } (i, j) \in \bar{W}_{pq}$$

$$I_1 = -1, \text{ if } (i, j) \in \underline{W}_{pq}$$

$$I_2 = +1, \text{ if } (i, j) \in \bar{W}_{rs}$$

$$I_2 = -1, \text{ if } (i, j) \in \underline{W}_{rs}$$

$$\Delta_{pq} (I_1 + I_2 K) = \text{resultant augmented flow in arc } (i, j).$$

In order to satisfy the capacity constraints,

$$0 \leq x_{ij} + \Delta_{pq} (I_1 + I_2 K) \leq b_{ij}$$

If $(I_1 + I_2^K) > 0$, then

$$\Delta_{pq} \leq \frac{(b_{ij} - x_{ij})}{(I_1 + I_2^K)}$$

If $(I_1 + I_2^K) < 0$, then

$$\Delta_{pq} \leq -\frac{x_{ij}}{(I_1 + I_2^K)}$$

In this manner Δ_{pq} is calculated for all arcs $(i,j) \in (W_{pq} \cup W_{rs})$. Therefore,

$$\Delta_{pq} = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left\{ \begin{array}{ll} \frac{x_{ij}}{(I_1 + I_2^K)} & \text{if } (I_1 + I_2^K) < 0, \\ \frac{(b_{ij} - x_{ij})}{(I_1 + I_2^K)} & \text{if } I_1 + I_2^K > 0 \end{array} \right.$$

$$\Delta_{pq} = \min. (\Delta_{pq}, \frac{(c_{ij} - c)}{(c_{pq} + Kc_{rs})})$$

Once Δ_{pq} is obtained, the flows are updated in W_{pq} and W_{rs} cycles. One of the arcs $(u,v) \in (W_{pq} \cup W_{rs})$ leaves at one of its bounds respectively. The dual variables, c_{ij} 's, d_{ij} 's, $\psi(i,j) \in L \cup U$, C , D and Z are updated respectively.

Arc (p,q) for the next iteration is obtained as follows:

- (a) If $(u,v) = (i,j) \in (W_{pq})$, set $(p,q) = (r,s)$.
- (b) If $(u,v) = (i,j) \in (\{W_{pq} \cap W_{rs}\} \cup W_{rs})$, (p,q) does not change.

If the arc (p, q) is at one of its bounds, reverse the orientation of (p, q) and set $\bar{c}_{pq} = -\bar{c}_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$. By selecting an arc $(r, s) \in S^0$ at each iteration and augmenting flows in W_{pq} , and W_{rs} , the algorithm moves along J, L shown in Fig. 3.3. If $(C_1, D_1) \in S$, then the algorithm obtains this solution which is an optimum solution and hence stops. If S^0 is empty then the current solution,

$$(C, D) \in B_2 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$$

or,

$$(C, D) \in B_3 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} < 0$$

according to the theorem 11. L in Fig. 3.3 represents this solution.

THEOREM 11: If $S^0 = \{\phi\}$ then the current solution $(C, D) \in B_2$ if $\bar{c}_{pq}/\bar{d}_{pq} > 0$ and $(C, D) \in B_3$ if $\bar{c}_{pq}/\bar{d}_{pq} \leq 0$.

Proof: The necessary and sufficient conditions for $(C, D) \in B_2$ are, the non-basic arcs $(i, j) \in L \cup U$ whose $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} \leq 0$, do not exist.

The necessary and sufficient conditions for $(C, D) \in B_3$ are, the non-basic arcs $(i, j) \in L \cup U$ whose $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} > 0$, do not exist.

Suppose, $\frac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$

111-1 ANDUH
87170

Arcs of the type $\bar{c}_{ij} \geq 0$ and $\bar{d}_{ij} < 0$ satisfy (3.16) and (3.17) if $\bar{c}_{pq} > 0$ and $\bar{d}_{pq} > 0$. Arcs of the type $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} = 0$ satisfy (3.16) and (3.17), if $\bar{c}_{pq} < 0$ and $\bar{d}_{pq} < 0$.

Since $S^0 = \{\emptyset\}$, the non-basic arcs (i, j) of the type $\bar{c}_{ij} \geq 0$ and $\bar{d}_{ij} \leq 0$ are absent. Therefore $(C, D) \in B_2$.

Suppose, $\bar{c}_{pq}/\bar{d}_{pq} \leq 0$

Arcs of the type $\bar{c}_{ij} \geq 0$ and $\bar{d}_{ij} > 0$ satisfy $\frac{\bar{d}_{pq}}{\bar{d}_{ij}} < 0$ and $\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{ij}} \times \bar{c}_{ij} > 0$ if $\bar{c}_{pq} > 0$ and $\bar{d}_{pq} < 0$. Since $S^0 = \{\emptyset\}$ the non-basic arcs (i, j) whose $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} > 0$, $\forall (i, j) \in L \cup U$ are absent. Hence $(C, D) \in B_3$.

QED

Now, the algorithm moves towards M along the curve B_2 . According to theorem 2, if $(C_1, D_1) \in R_2$ the $(C^*, D^*) \in B_2$ and $D^* \geq D_1$, and $C^* \leq C_1$, it is justified in moving along L, M for finding an optimum solution.

Similarly if $(C_1, D_1) \in R_3$, the algorithm traverses along I, F, H, Y and finds an optimum solution. If $(C_1, D_1) \in R_3$ and $D_1 \geq D_{\max}$, then the algorithm moves along B_4 and B_3 curves, and obtains an optimum solution.

3.6 Statement of Algorithm:

A formal statement of the algorithm organised in a manner suitable for computer implementation is given below:

Step 1: Set count to zero. Solve the minimum cost flow problem

with $\sum_{(i,j) \in A} c_{ij} x_{ij}$ as the objective function value.

Let (B, L, U) be the optimum basis structure obtained and X be the solution. Compute,

$$C = \sum_{(i,j) \in A} c_{ij} x_{ij}, \text{ and } D = \sum_{(i,j) \in A} d_{ij} x_{ij}.$$

(C, D) is the minimum cost and budget respectively.

Treating B as the basic tree, define the dual variables π_j^c 's and π_j^d 's as follows.

$$\pi_1^c = 0, \quad \pi_j^c = \pi_1^c + c_{ij}, \quad \forall (i,j) \in B$$

$$\pi_1^d = 0, \quad \pi_j^d = \pi_1^d + d_{ij}, \quad \forall (i,j) \in B$$

Define \bar{c}_{ij} and \bar{d}_{ij} for all non-basic arcs $(i,j) \in L \cup U$ as follows:

$$\bar{c}_{ij} = \begin{cases} \pi_i^c - \pi_j^c + c_{ij}, & \forall (i,j) \in L \\ \pi_j^c - \pi_i^c - c_{ij}, & \forall (i,j) \in U \end{cases}$$

$$\bar{d}_{ij} = \begin{cases} \pi_i^d - \pi_j^d + d_{ij}, & \forall (i,j) \in L \\ \pi_j^d - \pi_i^d - d_{ij}, & \forall (i,j) \in U \end{cases}$$

$$\text{Compute, } \alpha_1 = (C - C_1)$$

$$\alpha_2 = (D - D_1)$$

$$\beta_1 = (C_1 - C)$$

$$\beta_2 = (D_1 - D)$$

$$\alpha_1 = \begin{cases} \alpha_i, & \text{if } \alpha_i > 0, \quad \forall i = 1, 2 \\ 0, & \text{if } \alpha_1 \leq 0, \end{cases}$$

$$\beta_1 = \begin{cases} \beta_i, & \text{if } \beta_i > 0, \quad \forall i = 1, 2 \\ 0, & \text{if } \beta_1 \leq 0, \end{cases}$$

$$Z = w_1 c_1 + w_2 \alpha_2 + r_1 \beta_1 + r_2 \beta_2$$

If $D_1 \leq D$ then flag = -1, else flag = 1.

Set $Z' = Z$, $X' = X$, $C' = C$, $D' = D$ and go to Step 2.

Step 2: Let $S' = \{(i, j) \in L \cup U : c_{ij} \leq 0 \text{ and } \text{flag} * d_{ij} > 0\}$

If $S' = \emptyset$ then go to Step 3. Otherwise select an arc $(p, q) \in S'$ and enter it into the basis. Perform pivot iteration, update x_{ij} 's.

If $\text{flag} * D > \text{flag} * D_1$, then circulate Δ flow against the orientation of V_{pq} where $\Delta = \frac{D - D_1}{d_{pq}}$, and go to Step 7.

If $Z > Z'$ then X' is an optimum solution, Stop, otherwise, update (B, L, U) , π_j^c 's, π_j^d 's, set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 3: Let $S' = \{(i, j) \in L \cup U : c_{ij} > 0 \text{ and } \text{flag} * d_{ij} > 0\}$.

Define,

$$\mu_{ij} = \frac{c_{ij} * \text{flag}}{d_{ij}}$$

If $S' = \emptyset$ then set flag = -flag and go to Step 4.

Otherwise select an arc (p, q) , $\mu_{pq} = \min_{(i, j) \in S'} (\mu_{ij})$, and enter it into the basis. Perform the pivot iteration, update x_{ij} 's.

If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq} , where $\Delta = (C - C_1)/c_{pq}$. Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Augment Δ flow in W_{pq} where $\Delta = (C - C_1)/c_{pq}$.

If $\text{flag} * D > \text{flag} * D_1$, then circulate Δ flow against the orientation of W_{pq} where $\Delta = (D - D_1)/d_{pq}$, then check if $Z > Z'$ then X' is an optimum solution, stop.

Otherwise, go to Step 7.

If $Z > Z'$ then X' is an optimum solution, stop.

Otherwise update (B, L, U) , π_j^C 's, π_j^D 's, set $Z' = Z$, $X' = X$, $C' = C$, and $D' = D$. Repeat this step.

Step 4: Let $S' = \{(i, j) \in L \cup U : c_{ij} > 0 \text{ and } d_{ij} = 0\}$.

If $S' = \emptyset$ then go to Step 5. Otherwise select an arc $(p, q) \in S'$ and enter it into the basis. Perform the pivot iteration, update x_{ij} 's.

If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq} where $\Delta = (C - C_1)/c_{pq}$. X is an optimum solution, stop.

If $Z > Z'$ then X' is an optimum solution, stop.

Otherwise, update (B, L, U) , π_j^c 's, π_j^d 's. Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 5: Let $S' = \{(i, j) \in L \cup U : c_{ij} > 0 \text{ and } \text{flag} * d_{ij} > 0\}$.

Define, $\lambda_{ij} = c_{ij} * \text{flag} / d_{ij}$.

If $S' = \{\emptyset\}$ then X' is an optimum solution, stop.

Otherwise select an arc (p, q) $\lambda_{pq} = \max_{(i, j) \in S'} (\lambda_{ij})$ and enter it into the basis, go to Step 6.

Step 6: Perform the pivot iteration, update x_{ij} 's.

If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq} where $\Delta = (C - C_1)/c_{pq}$. Then if $Z > Z'$, X' is an optimum solution, stop. Otherwise X is an optimum solution, stop.

If $Z > Z'$ then X' is an optimum solution, stop.

Otherwise update (B, L, U) , π_j^c 's, π_j^d 's. Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 7: If count = 2 then set $Z' = Z$, $X' = X$, $C' = C$, $D' = D$ and go to Step 8.

If $C < C_1$, then X is an optimum solution, stop.

Otherwise the basis structure is $(T \cup \{(p, q)\}, L, U)$.

Define $S^0 = \{(i,j) \in L \cup U: \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and}$

$(\bar{c}_{pq} - (\bar{d}_{pq}/\bar{d}_{ij}) * c_{ij} > 0)\}$. If S^0 is empty then set count = count + 1, $\bar{c}_{pq} = -c_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$, status = -status and go to Step 7. Otherwise, select an arc $(r,s) \in S^0$ and enter it into the basis $(T \cup \{p,q\})$.

Two cycles are formed i.e. W_{pq} , W_{rs} . Define,

$l = -\bar{d}_{pq}/\bar{d}_{rs}$, set $T_{ij} = 0, \forall (i,j) \in A$.

Define,

$$T_{ij} = \begin{cases} T_{ij} + l, & \text{if } (i,j) \in \bar{W}_{pq} \\ T_{ij} - l, & \text{if } (i,j) \in \bar{W}_{pq} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij} + K, & \text{if } (i,j) \in \bar{W}_{rs} \\ T_{ij} - K, & \text{if } (i,j) \in \bar{W}_{rs} \end{cases}$$

$$\Delta = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left(-\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \right. \\ \left. \left(-\frac{b_{ij} - x_{ij}}{T_{ij}} \right), \text{ if } T_{ij} > 0 \right).$$

$$\Delta_{pq} = \min \left(\Delta, \frac{(c_{pq} - c)}{(c_{pq} + K * c_{rs})} \right)$$

Update Δ_{pq} in W_{pq} and $K \Delta_{pq}$ in W_{rs} .

Define $S^0 = \{(i,j) \in L \cup U : \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and } (c_{pq} - (\bar{d}_{pq}/\bar{d}_{ij}) * c_{ij} > 0)\}$. If S^0 is empty then set count = count + 1, $c_{pq} = -c_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$, status = ..status and go to Step 7. Otherwise, select an arc $(r,s) \in S^0$ and enter it into the basis $(T \cup \{p,q\})$.

Two cycles are formed i.e. W_{pq} , W_{rs} . Define,

$K = -\bar{d}_{pq}/\bar{d}_{rs}$, set $T_{ij} = 0, \forall (i,j) \in A$.

Define,

$$T_{ij} = \begin{cases} T_{ij} + 1, & \text{if } (i,j) \in W_{pq} \\ T_{ij} - 1, & \text{if } (i,j) \in W_{rs} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij} + K, & \text{if } (i,j) \in W_{rs} \\ T_{ij} - K, & \text{if } (i,j) \in W_{pq} \end{cases}$$

$$\Delta = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left(-\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \left(-\frac{b_{ij} - x_{ij}}{T_{ij}} \right) \text{ if } T_{ij} > 0 \right).$$

$$\Delta_{pq} = \min \left(\Delta, \frac{(c_{pq} - c_{rs})}{c_{pq} + K * c_{rs}} \right)$$

Update Δ_{pq} in W_{pq} and $K \Delta_{pq}$ in W_{rs} .

Update (B, L, U) , π_J^c 's, π_J^d 's.

The arc (p, q) for the next iteration is obtained as follows.

Let (u, v) be the leaving arc at its respective bound.

If $(u, v) = (i, j) \in W_{pq}$, set $(p, q) = (r, s)$

If $(u, v) = (i, j) \in (W_{pq} \cap W_{rs} \cup W_{rs})$, (p, q) does not change

If $x_{pq} = 0$ or $x_{pq} = b_{pq}$, then set $\bar{c}_{pq} = -c_{pq}$, $\bar{d}_{pq} = -d_{pq}$,
status = -status. Repeat this step.

Step 8: The current basis structure is $(T \cup \{(p, q)\}, L, U)$.

If $\bar{c}_{pq} = 0$ then the current solution X is optimum, stop.

If $\bar{c}_{pq} < 0$, then set $c_{pq} = -\bar{c}_{pq}$, $d_{pq} = -\bar{d}_{pq}$,
status = -status.

If $\bar{d}_{pq} > 0$, then set flag = 1 else set flag = -1,

Go to Step 6.

3.7 Numerical Example:

In this section a small-network flow problem is solved to illustrate the various steps of WGNF algorithm. The network is shown in Fig. 3.4. The numbers c_{ij} , b_{ij} , d_{ij} are indicated over each arc. An amount of 5 units of flow is sent from source 1 to sink 6. The aspiration levels and weights are specified as follows

$w_1 = 0.15$, $w_2 = 0.10$, $r_1 = 0.35$, $r_2 = 0.4$, $c_1 = 165$ and
 $D_1 = 125$.

The steps of the algorithm are summarized in Table 3.1. The \uparrow indicates the basic pivot arc and \downarrow indicates the basic arc leaving the basis. The graphs corresponding to the basis at various iterations are shown in Fig. 3.6. In these graphs, the basic pivot arc is drawn as dashed lines. The algorithm traced the path I, J, L, N as shown in Fig. 3.5 and obtained $C^* = 145$ and $D^* = 155$ as an optimum solution.

3.8 Mathematical Formulation of Interval Goal Network Flow Problem

The mathematical formulation of interval goal network flow problem is as follows:

$$\text{Min. } Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2 \quad (3.18)$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -v, & \text{if } i=1 \\ v, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (3.19)$$

$$0 < x_{ij} < b_{ij}, \forall (i,j) \in A \quad (3.20)$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1 \quad (3.21)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1 \quad (3.22)$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_3 - \alpha_3 = C_2 \quad (3.23)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_4 - \alpha_4 = D_2 \quad (3.24)$$

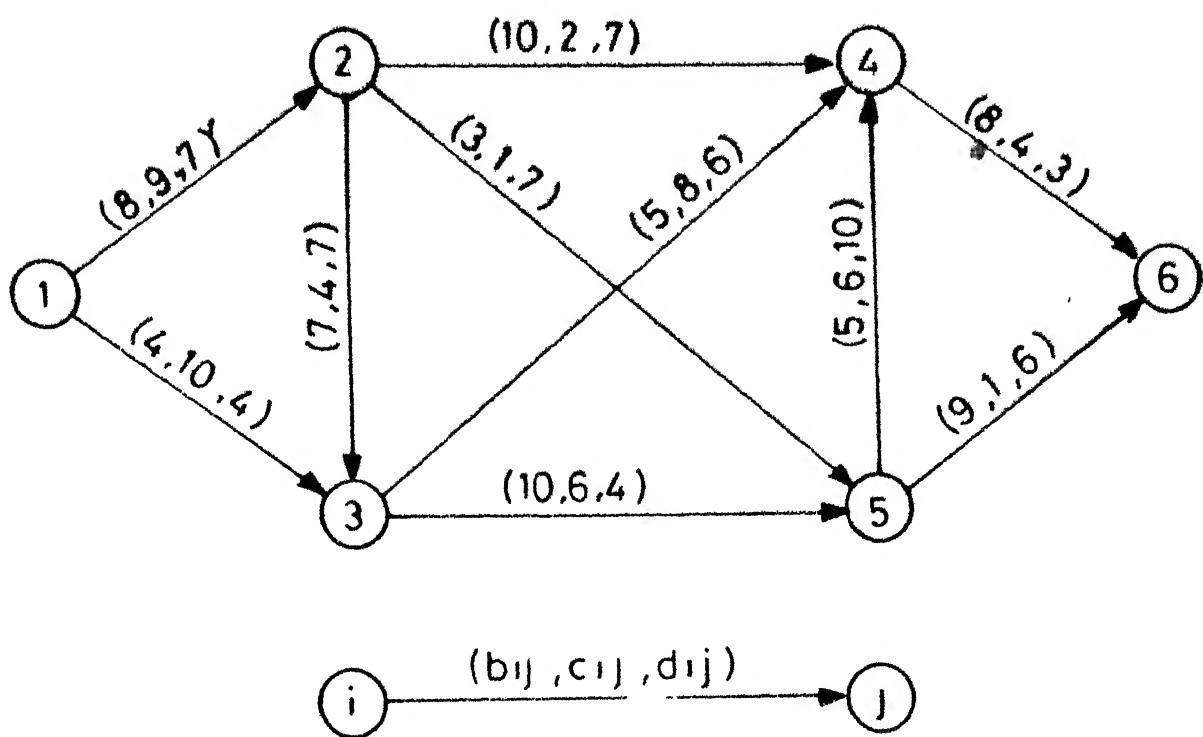


Fig. 3.4 Numerical example for the WLF problem

61 (a)

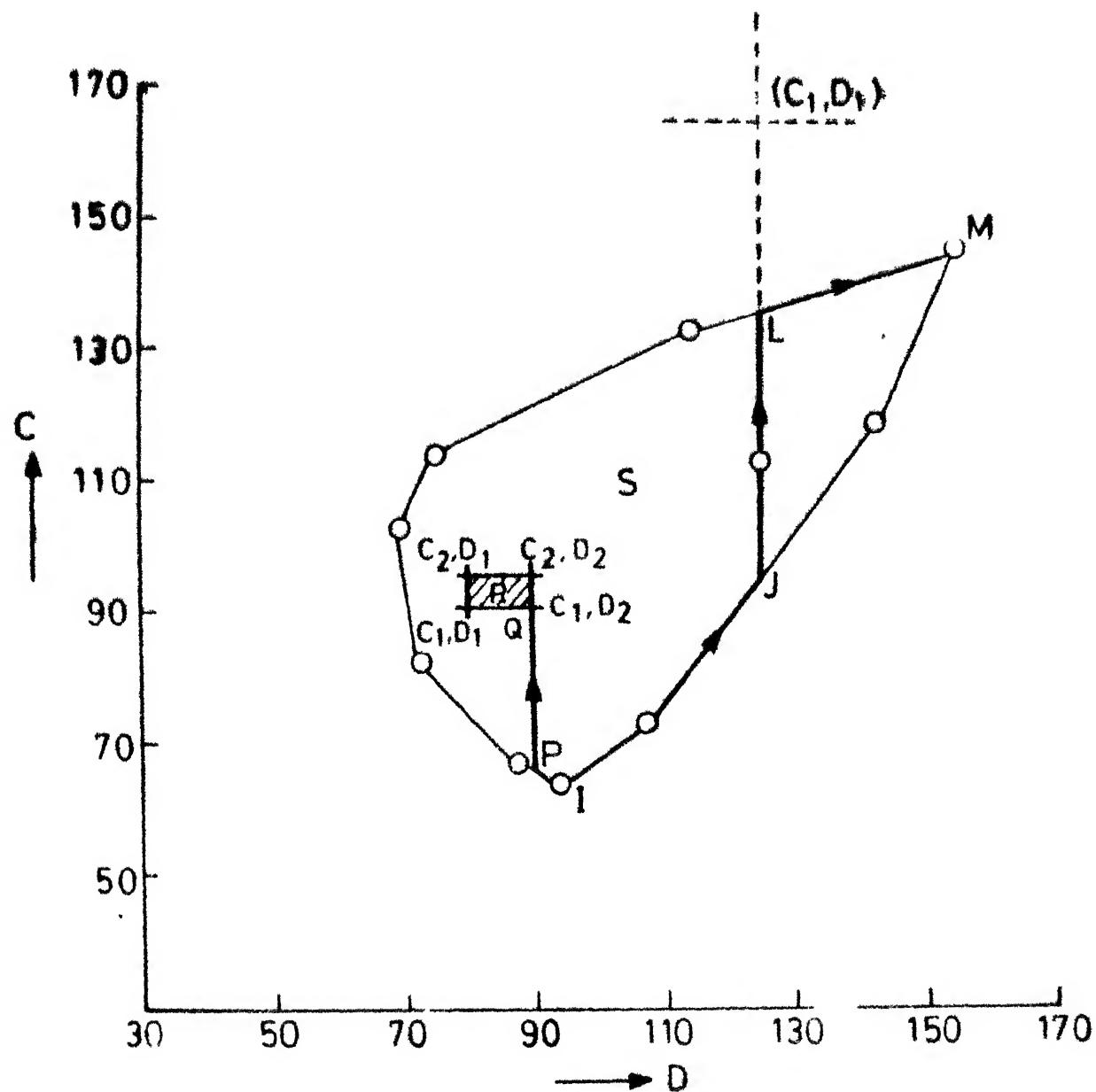


Fig. 3.5 Paths traced by WGNF algorithm and IGFN algorithm for the numerical example

61 (b)

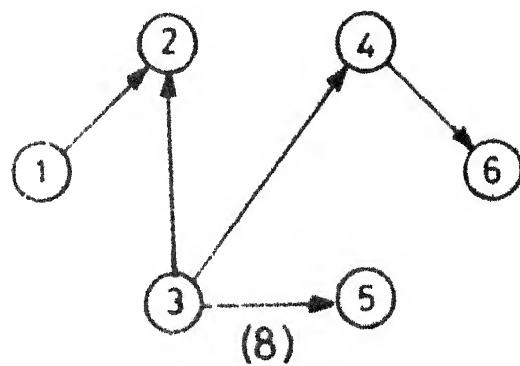
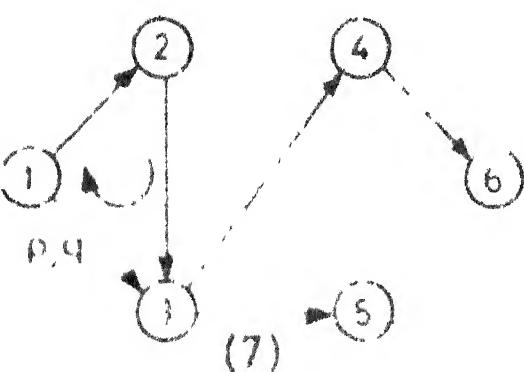
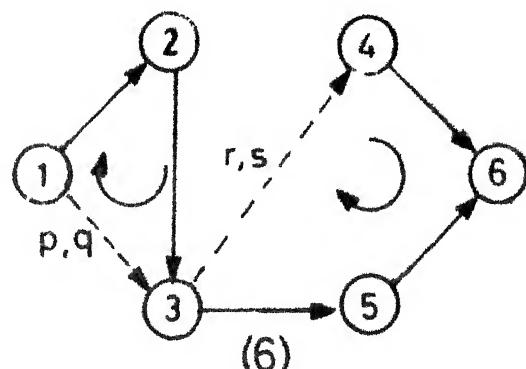
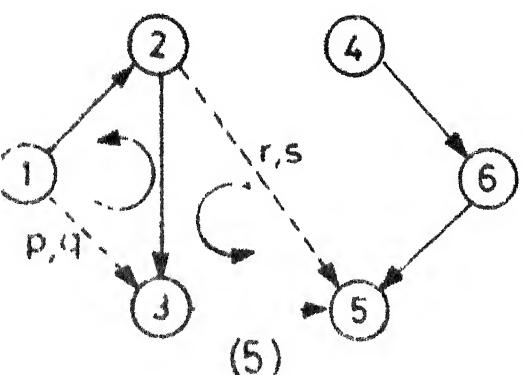
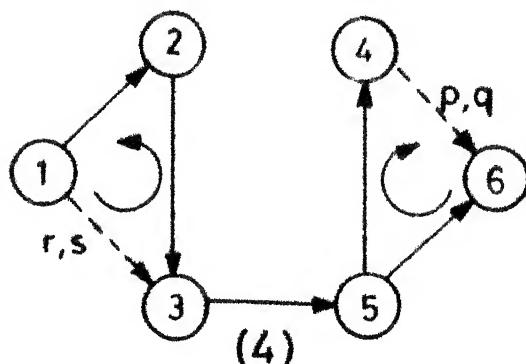
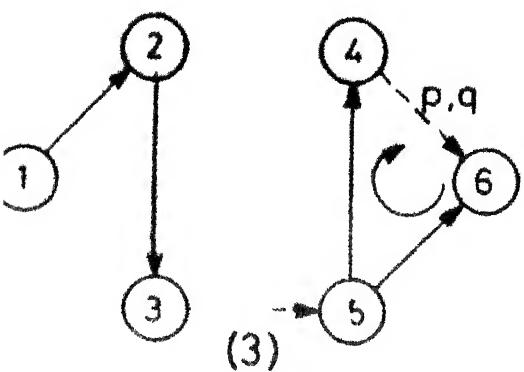
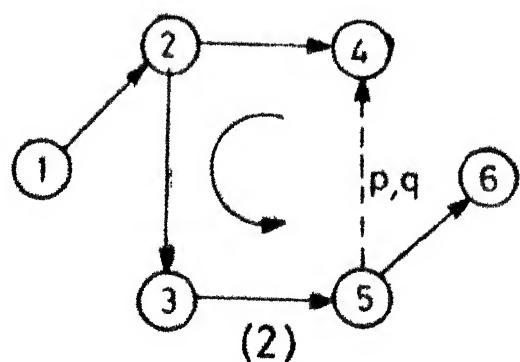
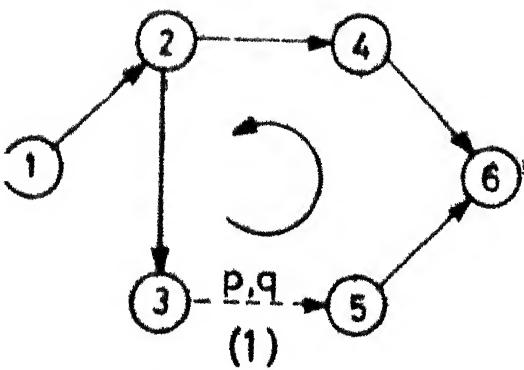


Fig 3.6 Basis in various iterations

Table 3.1: Solution of the WNE Problem.

C	D	Z	Basic Tree Arcs						Non-tree Arcs					
			(1,j)	(1,2)	(2,4)	(2,3)	(4,6)	(5,6)	(1,3)	(2,5)	(3,4)	(3,5)	(3,4)	(5,4)
0	63	94	$x_{1,j}$	5	2	3	2	3	0	3	0	0	0	0
			$\bar{c}_{1,j}$	0	0	0	0	0	-3	4	10	5	9	
			$\bar{d}_{1,j}$	0	0	0	0	0	-10	-3	6	7	7	$\uparrow(p,q)$
			(1,j)	(1,2)	(2,4)	(2,3)	(3,5)	(5,6)	(1,3)	(2,5)	(3,4)	(4,6)	(5,4)	
0.3	73	108	$x_{1,j}$	5	0	2	2	5	0	3	0	0	0	0
			$\bar{c}_{1,j}$	0	0	0	0	0	-3	9	10	-5	14	
			$\bar{d}_{1,j}$	0	0	0	0	0	-10	4	6	-7	14	$\uparrow(p,q)$
			(1,j)	(1,2)	(2,3)	(3,5)	(5,4)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(4,6)	
0.72	73	108	$x_{1,j}$	5	2	2	0	5	0	0	3	0	0	0
			$\bar{c}_{1,j}$	0	0	0	0	0	-3	-14	9	10	-5	
			$\bar{d}_{1,j}$	0	0	0	0	0	-10	-14	4	6	-7	$\uparrow(p,q)$
			(1,j)	(1,2)	(2,3)	(3,5)	(5,4)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(4,6)	
1.0	94.9	125	$x_{1,j}$	5	2	2	2.43	2.57	0	0	3	0	0	
			$\bar{c}_{1,j}$	0	0	0	0	0	-3	-14	9	10	-5	
			$\bar{d}_{1,j}$	0	0	0	0	0	-10	-14	4	6	-7	$\uparrow(p,q)$

		(1,2)	(1,3)	(4,6)	(3,5)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)	
		$x_{1,j}$	3.2	0.2	5.0	2.0	0	1.8	0	3	0	
		$\bar{c}_{1,j}$	0	0	0	0	0	-3	-5	9	5	
		$\bar{d}_{1,j}$	0	0	0	0	0	$\xrightarrow{0} (1,9)$	-7	$\xrightarrow{4} (1,8)$	-1	
112.6	125	(1,2)	(2,3)	(3,5)	(4,6)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)	
		$x_{1,j}$	2	2	5	5	0	3	0	0	0	
		$\bar{c}_{1,j}$	0	0	0	0	0	-3	-5	-9	5	
		$\bar{d}_{1,j}$	0	0	0	0	0	$\xrightarrow{10} (1,9)$	-10	-4	$\xrightarrow{7} (1,8)$	-1
136	125	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)	
		$x_{1,j}$	2	2	0	5	5	3	0	0	0	
		$\bar{c}_{1,j}$	0	0	0	0	0	3	-10	-9	-4	
		$\bar{d}_{1,j}$	0	0	0	0	0	$\xrightarrow{10} (1,9)$	-6	-4	-8	
136	125	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)	
		$x_{1,j}$	5	0	5	5	0	0	0	5	0	
		$\bar{c}_{1,j}$	0	0	0	0	0	-3	-10	-9	-4	
		$\bar{d}_{1,j}$	0	0	0	0	0	$\xrightarrow{10} (1,9)$	-6	-4	-8	
145	155	10.00	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	

where,

- C_1 = lower limit on total cost.
- C_2 = upper limit on total cost.
- D_1 = lower limit on total budget.
- D_2 = upper limit on total budget.

We are interested in a solution between C_1 and C_2 .

Therefore β_1 and α_3 are minimized. Similarly if the solution is to be between D_1 and D_2 , then β_2 and α_4 are minimized. By assigning proper weights, we obtain the objective function (3.18).

3.9 The Algorithm for Interval Goal Network Flow Problem:

The algorithm for interval goal network flow problem is similar to that of weighted goal network flow problem except for a few modifications.

The procedure to trace the path to reach an optimum solution is same as that of WGF algorithm but the paths that these two algorithms take to obtain the optimum solutions are different. In interval goal problem, the decision maker specifies a range of aspiration levels for each of the goals and weighting factors for deviations from the ranges. The ranges specified form a rectangular region R in E whose four corner points are (C_1, D_1) , (C_2, D_1) , (C_2, D_2) , (C_1, D_2) as shown in Fig. 3.7. The algorithm obtains an optimum solution $(x, \alpha_3, \alpha_4, \beta_1, \beta_2)$ by minimizing the sum of weighted deviations.

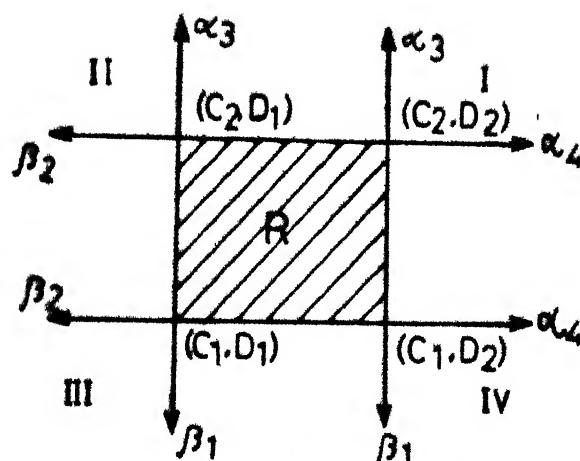


Fig. 3.7 Aspiration levels and deviations for IGFN problem

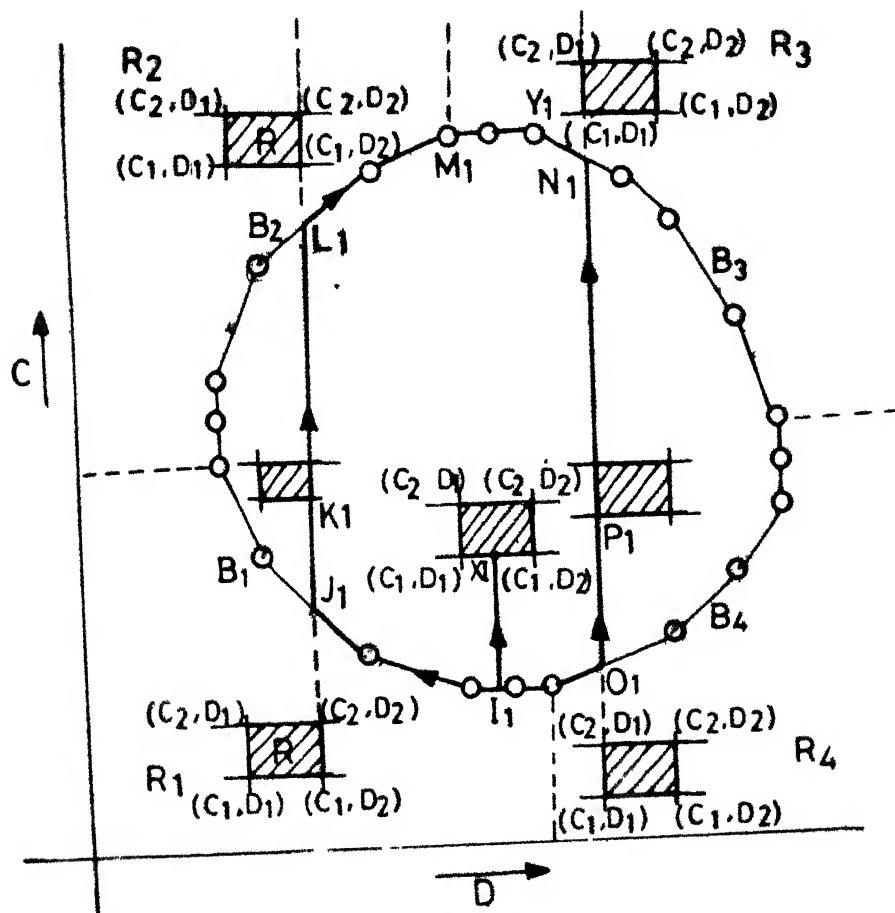


Fig 3.8 Various paths traced by IGFN algorithm to obtain optimum solutions

From above, it can be inferred that for different positions of R, the IGNF algorithm considers one of the corner co-ordinates of R as aspiration levels and obtains an optimum solution like the WGNF algorithm. Therefore, all the theorems proved for WGNF algorithm are also valid for IGNF algorithm.

3.10 Numerical Example:

In this section a small network flow problem is solved to illustrates the various steps of IGNF algorithm. The network is shown in Fig. 3.4. The aspiration level ranges and the weighting factors specified for this problem are as follows:
 $C_1 = 91, C_2 = 96, D_1 = 80, D_2 = 90, w_3 = 0.2, w_4 = 0.3,$
 $r_1 = 0.15, r_2 = 0.35.$

The steps of the algorithm are summarized in Table 3.2. The \uparrow indicates the basic pivot arc and \downarrow indicates the basic arc leaving the basis. The algorithm traces I,P,Q path as shown in Fig. 3.5 and obtains C = 91 and D = 90 as an optimum solution. Q represents an optimum solution, in Fig. 3.5.

3.11 Computational Results:

The algorithms proposed in this chapter were coded, debugged and tested in Fortran - 10 on DEC-1090 multi-programming, time sharing computer system. A number of randomly generated network problems were solved with different number

Table 3.2: Solution of the IGFN Problem.

It.	C	D	Z	Basic Tree Arcs						Non-tree Arcs			
				(1,2)	(1,3)	(2,4)	(4,6)	(5,6)	(2,5)	(2,3)	(3,4)	(3,5)	(5,4)
1	63	94	5	x_{ij}	5	0	2	3	2	3	0	0	0
				\bar{c}_{ij}	0	0	0	0	0	4	3	7	2
				\bar{d}_{ij}	0	0	0	0	0	-3	10	-4	9
				(i,j)	(1,2)	(1,3)	(2,4)	(5,6)	(4,6)	(2,5)	(2,3)	(3,4)	(3,5)
													(5,4)
2	0.67	65.7	90	x_{ij}	3.67	1.33	0.67	4.33	0.67	3	0	0	4.33
				\bar{c}_{ij}	0	0	0	0	0	4	1	-4	0
				\bar{d}_{ij}	0	0	0	0	0	-3	10	3	2
				(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)
													(5,4)
3	67.6	90	3.51	x_{ij}	3.2	1.8	0	2	5	3	0	0	0
				\bar{c}_{ij}	0	0	0	0	0	6	1	7	-2
				\bar{d}_{ij}	0	0	0	0	0	-6	10	-4	11
				(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)
													(5,4)
4	91	90	0	x_{ij}	2	3	0	5	5	0	2	0	0
				\bar{c}_{ij}	0	0	0	0	0	-6	1	7	-2
				\bar{d}_{ij}	0	0	0	0	0	6	10	-4	4

 $\uparrow(p, q)$

of nodes and arcs and computational times were noted. A network generator was used for computational study which generates well-structured networks. The program first generates a skeleton network for a specified width and length and then adds arcs randomly until the network contains a specified number of arcs.

Data structures based on augmented threaded index method[8] was implemented for storing the spanning tree. The tree was stored by means of thread indices, predecessor indices and the number of successors, each requiring an array of size n . Thread of a node i is a node that will be scanned in depth first search order of the sub-tree rooted at i . Predecessor of node i is the first node on the path from i to the root node of the tree. The number of successors of node i is the number of nodes in the subtree rooted at i . Link of a node i is the first arc on the path from i to the root node of the tree. If the arc is incident to i , it is stored as positive. If the arc is incident from i , it is stored as negative.

When an arc leaves the basis, it results in the formation of a hanging sub-tree. Thread indices help in scanning all the nodes of this sub-tree and thereby updating the dual variables. Predecessor indices are used for determining the minimum flow to be augmented, the leaving arc, and for updating the flows in the cycle formed, when a non-basic arc is added

to the basis. The number of successors for each node is maintained. It helps in determining the common node for the two paths when traversed from two ends of the entering arc towards the root node. Once the common node is known, it is easy to determine the minimum flow to be augmented in the cycle, leaving arc and updating the flow as well as the basis structure. Link is an array of size n that stores the arcs belonging to the basic tree. The storage for preserving the flows in arcs was reduced from m to n . This was achieved by storing the flows of basic arcs requiring an array of size n . The remaining non-basic arcs exist either at their lower or upper bounds, respectively. If the non-basic arc is at its upper bound, its capacity is made negative otherwise its capacity remains positive.

The main emphasis with computational results was laid on (i) to check the number of iterations performed by the algorithm and (ii) the computational time taken by the algorithms to get the optimum solutions. Problems sizes ranging from 10 nodes and 40 arcs to 200 nodes and 2500 arcs which includes both the sparse as well as dense networks were considered for computational study. Arc capacities were randomly generated between 5 to 100 whereas c_{ij} 's and d_{ij} 's were randomly generated between 1 to 50. Each problem was solved for three different aspiration levels and the computational times as well as the number of iterations are noted in Tables 3.3 and

3.4. About 20 problems of different sizes were solved for each of the algorithms. By comparing the columns (6), (8) and (10) of Table 3.3, we can infer that the time taken for solving $(C_1, D_1) \in R_2 \cup R_3$ is more than that for solving $(C_1, D_1) \in R_1 \cup R_4$ or $(C_1, D_1) \in S$. This is because, the optimum solution for $(C_1, D_1) \in R_2 \cup R_3$ lies on $B_2 \cup B_3$ and thereby traversing a longer path. No comparison can be made between (6) and (8) of Table 3.3 as the time taken to solve $(C_1, D_1) \in R_1 \cup R_4$ and $(C_1, D_1) \in S$ depends on the values of C_1 and D_1 . One can infer similar results by comparing columns (6), (8) and (10) of Table 3.4 of IGFN problem. The reason being that WGNF algorithm and IGFN algorithm behave in a similar manner.

It is evident from the tables that both the algorithms can solve quite large problems in reasonable amount of time. For instance, problem of size 400 nodes and 2500 arcs was solved in about $2\frac{1}{2}$ minutes by both the algorithms. Infact the time taken by both the algorithms are comparable because IGFN algorithm is similar to WGNF algorithm except for a few modifications which will not alter computational times significantly.

Both the algorithms suggested by us are able to solve practically large problems in a reasonable amount of computer time. From the literature review we can conclude that no work was done in the field of goal programming techniques applied to

Table: 3.3: Computational Times of WGNF Algorithm.
(Execution Times in Seconds on DEC-1090 System)

Width	Length	Nodes	Arcs	$(C_1, D_1) \in R_1$	$(C_1, D_1) \in S$	$(C_1, D_1) \in R_2$	$(C_1, D_1) \in R_3$
				ITER	TIME	ITER	TIME
3	3	10	40	36	0.07	34	0.053
3	5	15	70	82	0.17	102	0.14
5	5	25	100	106	0.27	151	0.23
5	5	25	200	84	0.34	71	0.24
5	10	50	200	282	1.45	369	1.56
5	10	50	400	141	1.17	100	0.65
5	15	75	300	435	3.22	386	1.24
5	15	75	600	230	2.74	277	3.21
10	10	100	400	580	5.73	1115	4.51
10	10	100	1000	313	6.1	303	5.35
						383	7.59

Table 3.3 continued

Width	Length	Nodes	Arcs	$(C_1, D_1) \in \mathcal{E}_1 \cup \mathcal{E}_4$		$(C_1, D_1) \in \mathcal{S}$		$(C_1, D_1) \in \mathcal{E}_2 \cup \mathcal{E}_3$	
				ITER	TIME	ITER	TIME	ITER	TIME
5	25	75	500	785	9.56	332	30.74	947	11.38
5	25	75	1200	1652	48.92	13370	99.64	3246	93.44
10	15	150	600	1062	15.42	1262	11.96	1369	19.59
10	15	150	1500	3226	118.45	4172	115.15	4334	155.5
7	25	175	700	1315	22.19	1711	23.28	1701	28.33
7	25	175	1600	3042	133.40	9480	147.19	4532	173.83
10	20	200	1000	403	8.02	672	8.93	915	20.40
10	20	200	2000	3674	146.39	9521	153.26	4756	181.57
20	20	400	1500	3709	139.37	9377	146.26	4375	173.69
20	20	400	2500	5213	172.56	11361	199.52	6857	208.18

Table 3.4: Computational Times of IGF Algorithm.
(Execution Times in Seconds on DEC-1090 Systems)

Width	Length	Nodes	Arcs	R \in P_1 UR ₄		R \in S		R \in P_2 UR ₂	
				ITER	TIME	ITER	TIME	ITER	TIME
3	3	10	40	36	0.07	34	0.053	58	0.11
3	5	15	70	82	0.17	102	0.139	82	0.18
5	5	25	100	114	0.32	124	0.22	167	0.47
5	5	25	200	88	0.39	136	0.36	191	0.95
5	10	50	200	243	1.26	541	1.86	391	2.03
5	10	50	400	158	1.37	610	1.91	379	2.71
5	15	75	300	397	2.93	302	1.14	535	3.98
5	15	75	600	237	2.89	280	3.07	602	8.40
10	10	100	400	607	6.09	410	2.16	791	7.78

Table 3.4 continued

Width	Length	Nodes	Arcs	R \in R ₁ UR ₄			R \in S			R \in R ₂ UR ₃		
				ITER	TIME	ITER	TIME	ITER	TIME	ITER	TIME	
10	10	100	1000	313	6.06	458	5.52	1139	12.35			
5	25	125	500	387	4.57	384	3.95	915	11.13			
5	25	125	1200	1603	47.89	11053	85.37	3085	89.20			
10	15	150	600	1096	16.05	2338	12.41	1374	19.72			
10	15	150	1500	3361	123.31	11944	125.61	4444	159.60			
7	25	175	700	1370	23.27	2695	28.91	1739	29.17			
7	25	175	1600	3535	138.60	13692	186.14	4633	177.83			
10	20	200	1000	426	8.648	1437	15.20	3507	26.31			
10	20	200	2000	3572	141.35	9407	150.82	4947	183.58			
20	20	400	1500	3657	138.07	9938	167.55	4489	179.56			
20	20	400	2500	5008	165.09	10021	195.83	6517	203.18			

network flow problems. This made us to explore this field, and suggest exact algorithms for weighted goal and interval goal network flow problems. One disadvantage with goal programming techniques is, the decision maker should specify proper weighting factors otherwise he may not obtain the preferred solution.

REFERENCES

1. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained minimum cost flow problem, Research paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).
2. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained maximum flow problem, working paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).
3. Ambrose, G., Hansen, D.R., and Lucien, D., Multi-objective Decision Analysis with Engineering and Business Applications, John Wiley and Sons, (1982).
4. Chen, S., and Saigal, R., A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints, Networks 7 (1977), 59-80.
5. Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J. (1963).
6. Glover, F., Network Applications in Industry and Government.
7. Ignizio, Jame, P., An Approach to the Modelling and Analysis of Multi-objective Generalised Networks, EJOR (Netherlands) 12 (1973) 4, 357-361.
8. Kennington, J.L., and Helgason, R.V., Algorithms for Networks Programming, John Wiley and Sons, Inc., New York, (1980).
9. Klingman, D., and Hultz, J., Solving Constrained Generalized Network Problems, Research Report CCS 257 Centre for Cybernetics Studies, University of Texas, Austin, Texas, Nov. 1976.
10. Klingman, D., and Mote, J., Solution Approaches for Network Flow Problems with Multiple Criteria, AMS (India) 1 (1972), Jan.
11. Lawler, E., Combinatorial Optimization, Networks and Matroids, Holt, Rinehart and Winston (1976).
12. Takashi, K., The Lexico-shortest Route Algorithm for Solving the Minimum Cost Flow Problem with an Additional Linear Constraint, Journal of Operations Research Society of Japan, Vol. 26, No. 3, Sept. 1983.
13. Zeleny, M., Multi-criteria Decision Making, McGraw-Hill Book Company, 1982.

```

0001 * PROGRAM TO SOLVE BI-CRITERIA MINIMUM COST FLOW PROBLEM BY
0002 * GOAL PROGRAMMING
0003 * ****
0004 * INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
0005 * FINISH,ENTER,V,STATUS,O,OP,Z,X,R,
0006 * TIME1,TIME2,TIME,CMIN,TR,
0007 * PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
0008 * SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
0009 * PG,RS,TAILPO,HEADPO,TAILRS,HEADRS,CODE,CODE1,CODE2,OVER
0010 * REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,MINIM,IFLOW
0011 * DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
0012 * COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
0013 * LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
0014 * LEAVE,RFLOW
0015 * TYPE 10
0016 * FORMAT(1X,'>>',*)
0017 * ACCEPT *,WIDTH,LENGTH,M,SEED,V,C1,D1
0018 * CALL SETRAN(SEED)
0019 * CALL NG4(WIDTH,LENGTH,N,M,SOURCE,SINK,POINT,HEAD)
0020 * DO 20 J=1,M
0021 *   CAP(J)=IRAN(5,100)
0022 *   COST(J)=IRAN(1,50)
0023 *   BUD(J)=IRAN(1,50)
0024 *   READ(21,*)W1,W2,W3,W4
0025 * ****
0026 * THIS PART OF PROGRAM FORMS THE BASIS TREE AND CALCULATES THE *
0027 * DUAL VARIABLES.
0028 * ****
0029 * CALL RTIME(TIME1)
0030 * COVER=0
0031 * S=1.0E-5
0032 * FLAG=0
0033 * RFLG=0
0034 * TFLG=0
0035 * DO 30 I=1,N
0036 *   PRED(I)=0
0037 *   PI(I)=0
0038 *   PID(I)=0
0039 * CONTINUE
0040 * LARGE=10000000
0041 * EQ=0
0042 * MINIM=1.0E+14
0043 * BUD(0)=0
0044 * ROOT=SOURCE
0045 * MA=M+1
0046 * CAP(MA)=LARGE
0047 * COST(MA)=LARGE
0048 * BUD(MA)=0
0049 * PRED(SINK)=SOURCE
0050 * LINK(SINK)=MA
0051 * PREV=SINK;TOP=1;COUNT=0
0052 * KOUNT=1
0053 * SCAN(TOP)=ROOT
0054 * IF(TOP.EQ.0) GO TO 70
0055 * MSGAN(TOP)
0056 * IF(I.GT.0) GO TO 50
0057 * I=1
0058 * NUMBER(I)=COUNT-LIST(I)
0059 * TOP=TOP-1
0060 * GO TO 40
0061 * LIST(I)=COUNT
0062 * COUNT=COUNT+1
0063 * THREAD(PREV)=I
0064 * PREV=I
0065 * SCAN(TOP)=-1
0066 * IF(COUNT.GE.N)GO TO 40
0067 * IX=POINT(I)
0068 * IY=POINT(I+1)-1
0069 * IF(IX.GT.IY) GO TO 40
0070 * DO 60 J=IX,IY
0071 *   K=HEAD(J)
0072 *   IF(PRED(K).NE.0) GO TO 60
0073 *   PRED(K)=I
0074 *   DIA(K)=J
0075 *   TOP=TOP+1
0076 *   SCAN(TOP)=K
0077 *   KOUNT=KOUNT+1
0078 *   CONTINUE
0079 *   GO TO 40
0080 * 70 THREAD(PREV)=SINK
0081 * PRED(SINK)=SOURCE
0082 * NUMBER(SINK)=1
0083 * NUMBER(SOURCE)=N
0084 * COUNT=1
0085 * 80 PRED(1)=0
0086 * DIA(1)=0
0087 * TOP=1
0088 * PRED(1)=1
0089 * COUNT=1
0090 * ****
0091 * END

```

```

90
I=SINK
K=READ()
PI(K)=PI(PRED(K))+COST(LINK(K))
PIO(K)=PI(PRED(K))+BUD(LINK(K))
FLD(K)=0
IF(K.EQ.PREV)GO TO 100
I=K
GO TO 90
100
CONTINUE
CODE=5
110
RSIGNE=1
DMINIM=0
DMINVIM=0
*****
* THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON - BASIC *
* ARC PO FOR FLOW AUGMENTATION. *
*****
120
SIGN=1
MUEMAX=LARGE
IF(C1.LT.DMINIM)SIGN=1
130
FLOWPO=0
ITER=ITER+1
IF(C1.EQ.0.AND.C.GE.C1)GO TO 920
IF(C1.EQ.0.AND.D.NE.D1.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 920
Q=ABS(D-D1)
IF((Q.DD.E.2).AND.(Q.LE.S))GO TO 500
ENTER=0
DO 230 I=1,N
START=POINT(I)
FINISH=POINT(I+1)-1
IF(START.GT.FINISH)GO TO 230
DO 230 J=START,FINISH
K=HEAD(J)
IF(CAP(J).LT.0)GO TO 140
CBAR=COST(J)+PI(I)-PI(K)
DBAR=BUD(J)+PID(I)-PID(K)
STATUS=1
GO TO 150
140
CBAR=PI(K)-PI(I)-COST(J)
DBAR=PID(K)-PID(I)-BUD(J)
STATUS=-1
150
GO TO(160,210,170,210,200)CODE
160
IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
GO TO 180
170
IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
180
ENTER=J
DMIN=CBAR
DMIN=DBAR
STAT90=STATUS
IHEAD=K
ITAIL=I
PO=J
GO TO 270
200
IF(CBAR.GE.0.OR.MUEMAX.LE.CBAR)GO TO 230
MUEMAX=CBAR
GO TO 220
210
IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
MUE=(RSIGNE*CBAR)/(SIGN*DBAR)
IF(MUE.GE.MUEMAX)GO TO 230
MUEMAX=MUE
220
ENTER=J
DMIN=CBAR
DMIN=DBAR
STATPO=STATUS
PO=J
IHEAD=K
ITAIL=I
230
CONTINUE
IF(CCODE.EQ.5.AND.ENTER.NE.0)GO TO 270
IF(CCODE.EQ.5.AND.ENTER.EQ.0)CODE=1
IF(ENTER.NE.0)GO TO 260
IF(CCODE.EQ.4.AND.ENTER.EQ.0)GO TO 920
CODE=CCODE+1
IF(CCODE.NE.4)GO TO 120
240
SIGN=RSIGNE
RSIGNE=-RSIGNE
250
MUEMAX=0
GO TO 130
260
MAX1=(C1-I)/DMIN
MAX2=(D1-I)/DMIN
MAX=MAX1
IF(MAX1.GT.MAX2.AND.MAX2.GT.0).OR.MAX1.LE.0)MAX=MAX2

```



```

      P1C1T1K1E2G2H1G2E2
      B002 E001K1S1T1U1D1, C1
      C0C1+ADD1
      D0D1+ADD2
      IF(ABS(C1).LE.S, AND, CODE.GE.3)GO TO 420
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,REFLAG)
      IF(OVER.EQ.1)GO TO 440
      DELTA=DELTA+MAX
      CALL AUGMEN
      GO TO 920
      DELTA=MAX
      C0C1+ADD1
      D0D1+ADD2
      CALL AUGMEN
      GO TO 920
      440  DELTA=DELTA1-MAX
      ADD=CMIN*STATUS*DELTA
      ADD1=DMIN*STATUS*DELTA
      C0C1+ADD
      D0D1+ADD1
      DELTA=DELTA1
      IF(DELTA1.LE.S)GO TO 480
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,REFLAG)
      C0C1+ADD
      D0D1+ADD1
      IF(OVER.EQ.1, AND, (DELTA.EQ.(DELTA1-MAX)))GO TO 440
      IF(OVER.EQ.1)GO TO 920
      C0C1+ADD
      D0D1+ADD1
      480  IF(LEG.EQ.0)CALL AUGMEN
      IF(LEG.NE.0)CALL UPDATE
      490  IF(CODE.EQ.4)GO TO 250
      GO TO 120
***** THIS PART OF THE PROGRAM SELECTS THE NON - BASIC ARC RS FOR *
* SIMULTANEOUS FLOW AUGMENTATION IN TWO CYCLES. *
***** 500  CBP0=CMIN
      DBP0=DMIN
      TAIP0=TAIL
      HEADP0=IHEAD
      FLOWP0=DELTA
      IF(STATP0.EQ.-1)FLWP0=-CAP(P0)-DELTA
      CODE2=1
      RS=D
      ITER=ITER+1
      DO 530 I=1,N
      SUM(I)=0.0
      IF(ABS(C-C1).LE.S)GO TO 920
      DO 560 I=1,N
      START=POINT(I)
      FINISH=POINT(I+1)-1
      IF(START.GT.FINISH)GO TO 560
      DO 560 J=START,FINISH
      K=HEAD(J)
      IF(CAP(J).LT.0)GO TO 540
      CBAR=COST(J)+PI(I)-PI(K)
      DBAR=BUD(J)+PID(I)-PID(K)
      STATUS=1
      GO TO 550
      540  CBAR=PI(K)-PI(I)-COST(J)
      DBAR=PID(K)-PID(I)-BUD(J)
      STATUS=-1
      550  IF(DBAR.EQ.0)GO TO 560
      IF(DBAR/DBP0.GE.0)GO TO 560
      IF((DBP0-(DBP0*CBAR)/DBAR).LE.0)GO TO 560
      RS=1
      TAILRS=1
      HEADRS=K
      STATRS=STATUS
      RATIO=-DBP0/DBAR
      CBR=CBAR
      DBRS=DBAR
      FLOWRS=0.0
      IF(STATRS.EQ.-1)FLOWRS=-CAP(RS)
      GO TO 590
      560  CONTINUE
      IF(CODE2.EQ.2, AND, RS.EQ.0)GO TO 570
      C0P0=-CBP0
      DBP0=-DBP0
      CAP(P0)=-CAP(P0)
      STATP0=-STATP0
      CODE3=CODE2+1
      GO TO 520
      570  C0P0=-CBP0
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,REFLAG)
      IF(C0P0.GE.0)GO TO 580
      C0P0=-CBP0
      DBP0=-DBP0
      CAP(P0)=-CAP(P0)

```

357 580 IF(CBPO.LT.0.AND.IDIR.EQ.-1)RFLAG=-1
 358 IF(CBPO.GT.0.AND.IDIR.EQ.1)RFLAG=+1
 359 IF(RFLAG.NE.0)SIGN=-SIGN
 360 FAAGE=1
 361 TELON=FLONPQ
 362 GO TO 240
 *** THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE
 * MTNIMUM FLOW TO BE AUGMENTED IN TWO CYCLES SIMULTANEOUSLY

 363 590 CODE1=1
 364 DELTA1=FLOWPQ
 365 IF(STATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
 366 LEG=0
 367 DELTA2=IABS(CAP(RS))/RATIO
 368 IF(CDELTA2.LT.DELTA1)DELTA1=DELTA2
 369 600 INTR=PQ
 370 STATUS=STATPQ
 371 RIN=1
 372 INC=1
 373 ITAIL=TAILPQ
 374 IHEAD=HEADPQ
 375 610 IF(STATUS.EQ.-1)GO TO 620
 376 IX=ITAIL
 377 IY=IHEAD
 378 GO TO 630
 379 620 IX=IHEAD
 380 IY=ITAIL
 381 630 IF(IX.EQ.IY)GO TO 790
 382 IF(NUMBER(IX).GT.NUMBER(IY))GO TO 710
 383 IF(CODE1.EQ.3)GO TO 660
 384 JE=LINK(IX)
 385 IF(J.LT.0)GO TO 640
 386 SUM(IX)=SUM(IX)+RIN
 387 GO TO 650
 388 640 SUM(IX)=SUM(IX)-RIN
 389 IF(CODE1.EQ.1)GO TO 700
 390 IF(SUM(IX)=670,700,680
 391 670 DELTA2=-FLOW(IX)/SUM(IX)
 392 GO TO 690
 393 680 JE=IABS(LINK(IX))
 394 DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
 395 690 IF(DELTA2.GE.DELTA1) GO TO 700
 396 DELTA1=DELTA2
 397 LEG=INC
 398 K1=IX
 399 TOTAL=FLOW(K1)+DELTA1*SUM(K1)
 400 700 IX=PREO(IX)
 401 GO TO 630
 402 710 IF(CODE1.GT.2)GO TO 740
 403 JE=LINK(IY)
 404 IF(J.LT.0)GO TO 720
 405 SUM(IY)=SUM(IY)-RIN
 406 GO TO 730
 407 720 SUM(IY)=SUM(IY)+RIN
 408 IF(CODE1.EQ.1)GO TO 780
 409 IF(SUM(IY)=50,780,760
 410 750 DELTA2=-FLOW(IY)/SUM(IY)
 411 GO TO 770
 412 760 JE=IABS(LINK(IY))
 413 DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
 414 770 IF(DELTA2.GE.DELTA1)GO TO 780
 415 DELTA1=DELTA2
 416 LEG=INC
 417 K1=IY
 418 TOTAL=FLOW(K1)+DELTA1*SUM(K1)
 419 780 IY=PREO(IY)
 420 GO TO 630
 421 790 IF(CODE1.EQ.1)IN1=IX
 422 IF(CODE1.EQ.2)IN2=IX
 423 CODE1=CODE1+1
 424 IF(CODE1.GT.3)GO TO 800
 425 IF(CODE1.EQ.3)GO TO 600
 426 ERIGERS
 427 ITAIL=ITAILRS
 428 IHEAD=HEADRS
 429 STATUS=STATRS
 430 RINRATIO
 431 INC2
 432 GO TO 610

433 *** THIS PART OF THE PROGRAM UPDATES THE FLOW IN BOTH THE CYCLES
 434 * BY CALLING IN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES
 435 * THE ARC TO
 436 * THE LEAVING ARC AND THE LEAVING FLOW
 437 * BY CALLING IN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES
 438 * THE ARC TO
 439 * THE LEAVING ARC AND THE LEAVING FLOW
 440 * BY CALLING IN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES
 441 * THE ARC TO
 442 * THE LEAVING ARC AND THE LEAVING FLOW
 443 * BY CALLING IN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES
 444 * THE ARC TO
 445 * THE LEAVING ARC AND THE LEAVING FLOW

800 DELTA2=(C1-C2)*(CBPQ+RATIO*CBRS)
 801 IF(DELTA2.GT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
 802 C1=CODE1*CBPQ*RATIO*DELTA1*CBRS
 803 C2=CODE1*CBPQ*RATIO*DELTA1*CBRS

```

PIUS1700E(GARGE
DELTA1=DELTA1*RATIO
ITAIL=TAILRS
IHEAD=HEADRS
STATUS=STATRS
FLOWRS=FLOWRS+STATRS*DELTA
IWE=1/2
ENTER=RS
CALL AUGMEN
DELTA1=DELTA1
ITAIL=TAILPO
IHEAD=HEADPO
STATUS=STATPO
FLOWPO=FLOWPO+STATPO*DELTA
IWE=1
ENTER=PO
CALL AUGMEN
IF( FLOWRS .NE. 0 .AND. FLOWRS .NE. IABS(CAP(RS))) GO TO 820
IF(ABS(FLOWRS) .GE. S .AND. ABS(FLOWRS-IABS(CAP(RS)))) GO TO 830
STATPO=-STATPO
CBPO=-CBPO
DBPO=-DBPO
810 IF( FLOWPO .LE. S .OR. ABS(FLOWPO-IABS(CAP(PQ))) .LE. S) GO TO 510
STATPO=-STATPO
CAP(PQ)=-CAP(PQ)
CBPO=-CBPO
DBPO=-DBPO
GO TO 510
820 FLOWPO=FLOWRS
CBPO=-CBRS
DBPO=-DBRS
POERS
TAILPO=TAILRS
HEADPO=HEADRS
830 STATUS=STATRS
IF( FLOWPO .LE. 0 .OR. ABS(FLOWPO-IABS(CAP(PQ))) .LE. S) GO TO 510
CAP(PQ)=-CAP(PQ)
STATPO=-STATPO
CBPO=-CBPO
DBPO=-DBPO
GO TO 510
840 IF( IABS(LEG) .EQ. 2) GO TO 880
ITAIL=TAILRS
IHEAD=HEADRS
STATUS=STATRS
DELTA1=DELTA1*RATIO
FLOWRS=FLOWRS+STATRS*DELTA
IWE=1/2
ENTER=RS
CALL AUGMEN
ITAIL=TAILPO
IHEAD=HEADPO
STATUS=STATPO
DELTA1=DELTA1
IWE=1/1
ENTER=PO
CMIN=CBPO
DMIN=SDBPO
IF(STATUS .EQ. 1) GO TO 850
CMIN=-CMIN
DMIN=-DMIN
850 ICAP=IABS(CAP(PQ))
K=K1
RFLW=FLOWPO
CALL UPDATE
CAP(LEAVE)=-IABS(CAP(LEAVE))
IF(TOTAL .LE. S) CAP(LEAVE)=-CAP(LEAVE)
FLOWPO=FLOWRS
POERS
TAILPO=TAILRS
HEADPO=HEADRS
CBPO=COST(PQ)+PI(TAILPO)-PI(HEADPO)
DBPO=BUD(PQ)+PID(TAILPO)-PID(HEADPO)
IF(STATRS .EQ. 1) GO TO 860
CBPO=-CBPO
DBPO=-DBPO
860 STATUSPO=STATRS
CBPO=-CBPO
DBPO=-DBPO
GO TO 830
870 ENTER=PO
ITAIL=TAILPO
IHEAD=HEADPO
STATUS=STATPO
DELTA1=DELTA1
IWE=1/1
FLOWPO=PI(TAILPO)+STATPO*DELTA1
CALL AUGMEN
END

```

```

PIESENKJEGARGE
DO IT BY YOURSELF

LINE OF HEADINGS
STATUS=STATRS
DELTA=DELTA1*RATIO
IWE=I42
LEG=LEG/2
ENTERRS
K=K1
CMIN=CBRS
DMIN=DBRS
IFCSTATUS, EQ.1 GO TO 890
CMIN=CMIN
DMIN=DMIN
890  IFCAP=1ABS(CAP(RS))
RELON=FLOWRS
CALL UPDATE
CAP(LEAVE)=-1ABS(CAP(LEAVE))
IF(PTOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
CBPO=CDST(P0)+PI(TAILP0)-PI(HEADP0)
DBPO=BDUD(P0)+PID(TAILP0)-PID(HEADP0)
IFCSTATPO, EQ.1 GO TO 900
CBPO=-CBPO
DBPO=-DBPO
900  STATPO=STATPO
CBPO=-CBPO
DBPO=-DBPO
910  GO TO 810
920  CALL RTIME(TIME2)
NRIPEC(22,930)C,D,TIME,ITER
930  FORKAT(1X,'COST=',F14.4,' BUDGET=',TIME=',I8,'ITER=',I8)
STOP
END

```


0001 0091 PITSOKI-EARGE
 0002 * THIS PROGRAM OBTAINS AN OPTIMUM SOLUTION FOR INTERVAL GOAL
 0003 * PROGRAMMING APPLIED TO BI-CRITERIA NETWORK FLOW PROBLEM.
 0004 *
 0005 * INTEGER WIDTH,SEED,ROOT, SOURCE, SINK, PREV,COUNT,FRONT,START,
 0006 * FINISH,ENTER,V,STATUS,Q,OP,2,X,R,
 0007 * TIME1,TIME2,TIME,CMIN,TR,
 0008 * PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
 0009 * SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
 0010 * P2,RS,TAILPO,HEADPO,TAILRS,HEADRS,CODE,CODE1,CODE2,OVER
 0011 * READ MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,MINIM,IFLOW
 0012 * DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
 0013 * COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,OMIN,IW,
 0014 * LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,E
 0015 * LEAVE,RFLOW
 0016 *
 0017 * TYPE 10
 0018 * FORMAT(1X,'>>\$)
 0019 * ACCEPT *,WIDTH,LENGTH,M,SEED,V,C1,D1,C2,D2
 0020 *
 0021 * ADD SETRAN(SEED)
 0022 *
 0023 * NG4(WIDTH,LENGTH,N,M,SOURCE,SINK,POINT,HEAD)
 0024 *
 0025 * DO 20 J=1,M
 0026 * ICAP(J)=IRAN(5,100)
 0027 * ICOST(J)=IRAN(1,500)
 0028 * BUD(J)=IRAN(1,500)
 0029 * READ(21,*)W1,W2,W3,W4
 0030 * CALL RTIME(TIME1)
 0031 *
 0032 * THIS PART OF THE PROGRAM FORMS THE BASIS TREE ALONG WITH THE
 0033 * PREDECESSOR,THREAD INDICES, NO. OF SUCCESSORS AND CALCULATES
 0034 * THE DUAL VARIABLES PI'S AND PID'S.
 0035 *
 0036 *
 0037 * OVER=0
 0038 * FLAG=0
 0039 * S=1.0E-5
 0040 * RFFLAG=0
 0041 * TFLAG=0
 0042 * DO 30 I=1,N
 0043 * PRED(I)=0
 0044 * PID(I)=0
 0045 * PID(I)=0
 0046 * CONTINUE
 0047 *
 0048 * LARGE=10000000
 0049 *
 0050 * MINIM=1.0E+14
 0051 * BUD(0)=0
 0052 * ROOT=SOURCE
 0053 * MA=M+1
 0054 * ICAP(MA)=LARGE
 0055 * ICOST(MA)=LARGE
 0056 * BUD(MA)=0.0
 0057 * PRED(SINK)=SOURCE
 0058 * LINK(SINK)=MA
 0059 * PREV=SINK;TOP=1;COUNT=0
 0060 * KOUNT=1
 0061 * SCAN(TOP)=ROOT
 0062 * IF(TOP.EQ.0) GO TO 70
 0063 *
 0064 * I=SCAN(TOP)
 0065 * IF(I.GT.0) GO TO 50
 0066 *
 0067 * I=NUMBER(I)=COUNT-LIST(I)
 0068 * TOP=TOP-1
 0069 * GO TO 40
 0070 *
 0071 * 50 LIST(I)=COUNT
 0072 * COUNT=COUNT+1
 0073 * THREAD(PREV)=I
 0074 * PREV=I
 0075 * SCAN(TOP)=I
 0076 * IF(KOUNT.GE.N)GO TO 40
 0077 * IX=PDINT(I)
 0078 * TYPEPOINT(I+1)=1
 0079 * IF(IX.GT.IY) GO TO 40
 0080 * DO 60 J=IX,IY
 0081 * K=HEAD(J)
 0082 * IF(PRED(K).NE.0) GO TO 60
 0083 * PRED(K)=I
 0084 * LINK(K)=J
 0085 * TOP=TOP+1
 0086 * SCAN(TOP)=K
 0087 * KOUNT=KOUNT+1
 0088 *
 0089 * 60 CONTINUE
 0090 * GO TO 40
 0091 *
 0092 * 70 THREAD(PREV)=SINK
 0093 * TURB(0)=SOURCE
 0094 * NUMBER(SINK)=1
 0095 * NUMBER(SOURCE)=N
 0096 *
 0097 * 80 TIME
 0098 * CTIME=0
 0099 * DTIME=0
 0100 *
 0101 *
 0102 *
 0103 *
 0104 *
 0105 *
 0106 *
 0107 *
 0108 *
 0109 *
 0110 *
 0111 *
 0112 *
 0113 *
 0114 *
 0115 *
 0116 *
 0117 *
 0118 *
 0119 *
 0120 *
 0121 *
 0122 *
 0123 *
 0124 *
 0125 *
 0126 *
 0127 *
 0128 *
 0129 *
 0130 *
 0131 *
 0132 *
 0133 *
 0134 *
 0135 *
 0136 *
 0137 *
 0138 *
 0139 *
 0140 *
 0141 *
 0142 *
 0143 *
 0144 *
 0145 *
 0146 *
 0147 *
 0148 *
 0149 *
 0150 *
 0151 *
 0152 *
 0153 *
 0154 *
 0155 *
 0156 *
 0157 *
 0158 *
 0159 *
 0160 *
 0161 *
 0162 *
 0163 *
 0164 *
 0165 *
 0166 *
 0167 *
 0168 *
 0169 *
 0170 *
 0171 *
 0172 *
 0173 *
 0174 *
 0175 *
 0176 *
 0177 *
 0178 *
 0179 *
 0180 *
 0181 *
 0182 *
 0183 *
 0184 *
 0185 *
 0186 *
 0187 *
 0188 *
 0189 *
 0190 *
 0191 *
 0192 *
 0193 *
 0194 *
 0195 *
 0196 *
 0197 *
 0198 *
 0199 *
 0200 *
 0201 *
 0202 *
 0203 *
 0204 *
 0205 *
 0206 *
 0207 *
 0208 *
 0209 *
 0210 *
 0211 *
 0212 *
 0213 *
 0214 *
 0215 *
 0216 *
 0217 *
 0218 *
 0219 *
 0220 *
 0221 *
 0222 *
 0223 *
 0224 *
 0225 *
 0226 *
 0227 *
 0228 *
 0229 *
 0230 *
 0231 *
 0232 *
 0233 *
 0234 *
 0235 *
 0236 *
 0237 *
 0238 *
 0239 *
 0240 *
 0241 *
 0242 *
 0243 *
 0244 *
 0245 *
 0246 *
 0247 *
 0248 *
 0249 *
 0250 *
 0251 *
 0252 *
 0253 *
 0254 *
 0255 *
 0256 *
 0257 *
 0258 *
 0259 *
 0260 *
 0261 *
 0262 *
 0263 *
 0264 *
 0265 *
 0266 *
 0267 *
 0268 *
 0269 *
 0270 *
 0271 *
 0272 *
 0273 *
 0274 *
 0275 *
 0276 *
 0277 *
 0278 *
 0279 *
 0280 *
 0281 *
 0282 *
 0283 *
 0284 *
 0285 *
 0286 *
 0287 *
 0288 *
 0289 *
 0290 *
 0291 *
 0292 *
 0293 *
 0294 *
 0295 *
 0296 *
 0297 *
 0298 *
 0299 *
 0300 *
 0301 *
 0302 *
 0303 *
 0304 *
 0305 *
 0306 *
 0307 *
 0308 *
 0309 *
 0310 *
 0311 *
 0312 *
 0313 *
 0314 *
 0315 *
 0316 *
 0317 *
 0318 *
 0319 *
 0320 *
 0321 *
 0322 *
 0323 *
 0324 *
 0325 *
 0326 *
 0327 *
 0328 *
 0329 *
 0330 *
 0331 *
 0332 *
 0333 *
 0334 *
 0335 *
 0336 *
 0337 *
 0338 *
 0339 *
 0340 *
 0341 *
 0342 *
 0343 *
 0344 *
 0345 *
 0346 *
 0347 *
 0348 *
 0349 *
 0350 *
 0351 *
 0352 *
 0353 *
 0354 *
 0355 *
 0356 *
 0357 *
 0358 *
 0359 *
 0360 *
 0361 *
 0362 *
 0363 *
 0364 *
 0365 *
 0366 *
 0367 *
 0368 *
 0369 *
 0370 *
 0371 *
 0372 *
 0373 *
 0374 *
 0375 *
 0376 *
 0377 *
 0378 *
 0379 *
 0380 *
 0381 *
 0382 *
 0383 *
 0384 *
 0385 *
 0386 *
 0387 *
 0388 *
 0389 *
 0390 *
 0391 *
 0392 *
 0393 *
 0394 *
 0395 *
 0396 *
 0397 *
 0398 *
 0399 *
 0400 *
 0401 *
 0402 *
 0403 *
 0404 *
 0405 *
 0406 *
 0407 *
 0408 *
 0409 *
 0410 *
 0411 *
 0412 *
 0413 *
 0414 *
 0415 *
 0416 *
 0417 *
 0418 *
 0419 *
 0420 *
 0421 *
 0422 *
 0423 *
 0424 *
 0425 *
 0426 *
 0427 *
 0428 *
 0429 *
 0430 *
 0431 *
 0432 *
 0433 *
 0434 *
 0435 *
 0436 *
 0437 *
 0438 *
 0439 *
 0440 *
 0441 *
 0442 *
 0443 *
 0444 *
 0445 *
 0446 *
 0447 *
 0448 *
 0449 *
 0450 *
 0451 *
 0452 *
 0453 *
 0454 *
 0455 *
 0456 *
 0457 *
 0458 *
 0459 *
 0460 *
 0461 *
 0462 *
 0463 *
 0464 *
 0465 *
 0466 *
 0467 *
 0468 *
 0469 *
 0470 *
 0471 *
 0472 *
 0473 *
 0474 *
 0475 *
 0476 *
 0477 *
 0478 *
 0479 *
 0480 *
 0481 *
 0482 *
 0483 *
 0484 *
 0485 *
 0486 *
 0487 *
 0488 *
 0489 *
 0490 *
 0491 *
 0492 *
 0493 *
 0494 *
 0495 *
 0496 *
 0497 *
 0498 *
 0499 *
 0500 *
 0501 *
 0502 *
 0503 *
 0504 *
 0505 *
 0506 *
 0507 *
 0508 *
 0509 *
 0510 *
 0511 *
 0512 *
 0513 *
 0514 *
 0515 *
 0516 *
 0517 *
 0518 *
 0519 *
 0520 *
 0521 *
 0522 *
 0523 *
 0524 *
 0525 *
 0526 *
 0527 *
 0528 *
 0529 *
 0530 *
 0531 *
 0532 *
 0533 *
 0534 *
 0535 *
 0536 *
 0537 *
 0538 *
 0539 *
 0540 *
 0541 *
 0542 *
 0543 *
 0544 *
 0545 *
 0546 *
 0547 *
 0548 *
 0549 *
 0550 *
 0551 *
 0552 *
 0553 *
 0554 *
 0555 *
 0556 *
 0557 *
 0558 *
 0559 *
 0560 *
 0561 *
 0562 *
 0563 *
 0564 *
 0565 *
 0566 *
 0567 *
 0568 *
 0569 *
 0570 *
 0571 *
 0572 *
 0573 *
 0574 *
 0575 *
 0576 *
 0577 *
 0578 *
 0579 *
 0580 *
 0581 *
 0582 *
 0583 *
 0584 *
 0585 *
 0586 *
 0587 *
 0588 *
 0589 *
 0590 *
 0591 *
 0592 *
 0593 *
 0594 *
 0595 *
 0596 *
 0597 *
 0598 *
 0599 *
 0600 *
 0601 *
 0602 *
 0603 *
 0604 *
 0605 *
 0606 *
 0607 *
 0608 *
 0609 *
 0610 *
 0611 *
 0612 *
 0613 *
 0614 *
 0615 *
 0616 *
 0617 *
 0618 *
 0619 *
 0620 *
 0621 *
 0622 *
 0623 *
 0624 *
 0625 *
 0626 *
 0627 *
 0628 *
 0629 *
 0630 *
 0631 *
 0632 *
 0633 *
 0634 *
 0635 *
 0636 *
 0637 *
 0638 *
 0639 *
 0640 *
 0641 *
 0642 *
 0643 *
 0644 *
 0645 *
 0646 *
 0647 *
 0648 *
 0649 *
 0650 *
 0651 *
 0652 *
 0653 *
 0654 *
 0655 *
 0656 *
 0657 *
 0658 *
 0659 *
 0660 *
 0661 *
 0662 *
 0663 *
 0664 *
 0665 *
 0666 *
 0667 *
 0668 *
 0669 *
 0670 *
 0671 *
 0672 *
 0673 *
 0674 *
 0675 *
 0676 *
 0677 *
 0678 *
 0679 *
 0680 *
 0681 *
 0682 *
 0683 *
 0684 *
 0685 *
 0686 *
 0687 *
 0688 *
 0689 *
 0690 *
 0691 *
 0692 *
 0693 *
 0694 *
 0695 *
 0696 *
 0697 *
 0698 *
 0699 *
 0700 *
 0701 *
 0702 *
 0703 *
 0704 *
 0705 *
 0706 *
 0707 *
 0708 *
 0709 *
 0710 *
 0711 *
 0712 *
 0713 *
 0714 *
 0715 *
 0716 *
 0717 *
 0718 *
 0719 *
 0720 *
 0721 *
 0722 *
 0723 *
 0724 *
 0725 *
 0726 *
 0727 *
 0728 *
 0729 *
 0730 *
 0731 *
 0732 *
 0733 *
 0734 *
 0735 *
 0736 *
 0737 *
 0738 *
 0739 *
 0740 *
 0741 *
 0742 *
 0743 *
 0744 *
 0745 *
 0746 *
 0747 *
 0748 *
 0749 *
 0750 *
 0751 *
 0752 *
 0753 *
 0754 *
 0755 *
 0756 *
 0757 *
 0758 *
 0759 *
 0760 *
 0761 *
 0762 *
 0763 *
 0764 *
 0765 *
 0766 *
 0767 *
 0768 *
 0769 *
 0770 *
 0771 *
 0772 *
 0773 *
 0774 *
 0775 *
 0776 *
 0777 *
 0778 *
 0779 *
 0780 *
 0781 *
 0782 *
 0783 *
 0784 *
 0785 *
 0786 *
 0787 *
 0788 *
 0789 *
 0790 *
 0791 *
 0792 *
 0793 *
 0794 *
 0795 *
 0796 *
 0797 *
 0798 *
 0799 *
 0800 *
 0801 *
 0802 *
 0803 *
 0804 *
 0805 *
 0806 *
 0807 *
 0808 *
 0809 *
 0810 *
 0811 *
 0812 *
 0813 *
 0814 *
 0815 *
 0816 *
 0817 *
 0818 *
 0819 *
 0820 *
 0821 *
 0822 *
 0823 *
 0824 *
 0825 *
 0826 *
 0827 *
 0828 *
 0829 *
 0830 *
 0831 *
 0832 *
 0833 *
 0834 *
 0835 *
 0836 *
 0837 *
 0838 *
 0839 *
 0840 *
 0841 *
 0842 *
 0843 *
 0844 *
 0845 *
 0846 *
 0847 *
 0848 *
 0849 *
 0850 *
 0851 *
 0852 *
 0853 *
 0854 *
 0855 *
 0856 *
 0857 *
 0858 *
 0859 *
 0860 *
 0861 *
 0862 *
 0863 *
 0864 *
 0865 *
 0866 *
 0867 *
 0868 *
 0869 *
 0870 *
 0871 *
 0872 *
 0873 *
 0874 *
 0875 *
 0876 *
 0877 *
 0878 *
 0879 *
 0880 *
 0881 *
 0882 *
 0883 *
 0884 *
 0885 *
 0886 *
 0887 *
 0888 *
 0889 *
 0890 *
 0891 *
 0892 *
 0893 *
 0894 *
 0895 *
 0896 *
 0897 *
 0898 *
 0899 *
 0900 *
 0901 *
 0902 *
 0903 *
 0904 *
 0905 *
 0906 *
 0907 *
 0908 *
 0909 *
 0910 *
 0911 *
 0912 *
 0913 *
 0914 *
 0915 *
 0916 *
 0917 *
 0918 *
 0919 *
 0920 *
 0921 *
 0922 *
 0923 *
 0924 *
 0925 *
 0926 *
 0927 *
 0928 *
 0929 *
 0930 *
 0931 *
 0932 *
 0933 *
 0934 *
 0935 *
 0936 *
 0937 *
 0938 *
 0939 *
 0940 *
 0941 *
 0942 *
 0943 *
 0944 *
 0945 *
 0946 *
 0947 *
 0948 *
 0949 *
 0950 *
 0951 *
 0952 *
 0953 *
 0954 *
 0955 *
 0956 *
 0957 *
 0958 *
 0959 *
 0960 *
 0961 *
 0962 *
 0963 *
 0964 *
 0965 *
 0966 *
 0967 *
 0968 *
 0969 *
 0970 *
 0971 *
 0972 *
 0973 *
 0974 *
 0975 *
 0976 *
 0977 *
 0978 *
 0979 *
 0980 *
 0981 *
 0982 *
 0983 *
 0984 *
 0985 *
 0986 *
 0987 *
 0988 *
 0989 *
 0990 *
 0991 *
 0992 *
 0993 *
 0994 *
 0995 *
 0996 *
 0997 *
 0998 *
 0999 *
 0900 *
 0901 *
 0902 *
 0903 *
 0904 *
 0905 *
 0906 *
 0907 *
 0908 *
 0909 *
 0910 *
 0911 *
 0912 *
 0913 *
 0914 *
 0915 *
 0916 *
 0917 *
 0918 *
 0919 *
 0920 *
 0921 *
 0922 *
 0923 *
 0924 *
 0925 *
 0926 *
 0927 *
 0928 *
 0929 *
 0930 *
 0931 *
 0932 *
 0933 *
 0934 *
 0935 *
 0936 *
 0937 *
 0938 *
 0939 *
 0940 *
 0941 *
 0942 *
 0943

```

PI(CST)K=EGAR0
PLOC(CST)K=0
FLDN(I)=0
FLDN(SINK)=V
>=V*LARGE
DED=3
I=SINK
90 K=PHREAD(I)
PI(K)=PI(PRED(K))+COST(LINK(K))
PID(K)=PID(PRED(K))+BUD(LINK(K))
FLDN(K)=0
IF(K.EQ.PREV)GO TO 100
I=K
GO TO 90
100 CONTINUE
CODE=5
110 RSIGN=-1
DMINIM=2
DMINIM=0
120 SIGN=-1
***** THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON-BASIC ARC PO ****
* FROM A SET OF NON-BASIC ARCS FOR FLOW AUGUMENTATION. *
***** **** **** **** **** **** **** **** **** **** **** **** **** **** **** **** ****
MUEMAX=LARGE
IF(D2.LT.DMINIM)SIGN=1
130 FLOWPO=0
IF(DMINIM.GE.D1.AND.DMINIM.LE.D2.AND.C.GE.C1.AND.CODE.NE.5)GO TO
180
IF((ABS(D-D1).LE.S.OR.ABS(D2-D).LE.S).AND.CODE.LE.2)GO TO 460
IF((ABS(C-C1).LE.S.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 880
IF((ABS(D-D1).LE.S.OR.ABS(D-D2).LE.S).AND.(C.GE.C1)
AND.(CODE.LE.2))GO TO 880
IF(D.GE.D1.AND.D.LE.D2.AND.C.GE.C1.AND.C.LE.C2)GO TO 880
ENTERD0
ITER=ITER+1
DO 230 I=1,N
START=POINT(I)
FINISH=POINT(I+1)-1
IF(START.GT.FINISH)GO TO 230
DO 230 J=START,FINISH
K=HEAD(J)
IF(CAP(J).LT.0)GO TO 140
CBAR=COST(J)+PI(I)-PI(K)
DBAR=BUD(J)+PID(I)-PID(K)
STATUS=1
GO TO 150
140 CBAR=PI(K)-PI(I)-COST(J)
DBAR=PID(K)-PID(I)-BUD(J)
STATUS=-1
150 GO TO(160,210,170,210,200)CODE
160 IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
GO TO 180
170 IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
180 ENTER=J
DMIN=CBAR
DMIN=0
STATPO=STATUS
IHEAD=K
ITAIL=I
POBJ
190 GO TO 270
200 IF(CBAR.GE.0.OR.MUEMAX.LE.CBAR)GO TO 230
MUEMAX=CBAR
GO TO 220
210 IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
MUE=(SIGN*CBAR)/(SIGN*DBAR)
IF(MUE.GE.MUEMAX)GO TO 230
MUEMAX=MUE
220 ENTER=J
CONTN=CBAR
DMIN=0
STATPO=STATUS
POBJ
IHEAD=K
ITAIL=I
CONTINUE
IF(CODE.EQ.5.AND.ENTER.NE.0)GO TO 270
IF(CODE.EQ.5.AND.ENTER.EQ.0)CODE=1
IF(ENTER.EQ.0)GO TO 250
IF(CODE.EQ.1.AND.ENTER.EQ.0)GO TO 880
CODE=CODE+1
IF(CODE.EQ.3.AND.C.LE.C2.AND.CODE.EQ.3)GO TO 880
IF(CODE.EQ.4)GO TO 120
240 SIGN=-SIGN
RSIGN=-RSIGN
250 AND=1
260 CONTINUE
270 MUE=(C2-C1)/CIN

```

```

IF(CODE.GT.3)MAX1=(C1-G)/CMIN
MAX2=(D1-D)/DMIN
IF(D2.LT.DMIN)MAX2=(D2-D)/DMIN
MAX=MAX1
IF((MAX1.GT.MAX2.AND.MAX2.GT.0).OR.(MAX1.LT.0))MAX=MAX2
*** THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC FROM THE CYCLE
* AND THE REVIVING FLOW TO BE AUGMENTED IN THE CYCLE.
*** AND THE REVIVING FLOW TO BE AUGMENTED IN THE CYCLE.
270 ICAP=CAP(ENTER)
LEG=0
FLOWPO=IABS(CAP(P0))
IF(SPATPO.EQ.1)FLOWPO=0
IF(FLAG.EQ.0)GO TO 280
FLDAPO=TFLOW
FLAG=0
280 IF(ICAP.LT.0)GO TO 290
STATUS=1
IX=ITAIL
IY=IHEAD
DELTAI1=ICAP-FLOWPO
GO TO 300
290 ICAP=-ICAP
CMIN=-CMIN
DMIN=-DMIN
STATUS=-1
IX=IHEAD
IY=ITAIL
DELTAI1=FLOWPO
DELTAI=DELTAI1
REFLOW=FLOWPO
310 IF(IX.EQ.IY)GO TO 390
IF(NUMBER(IX).GT.NUMBER(IY))GO TO 350
J=LINK(IX)
IF(J.LT.0)GO TO 320
CHANGE=CAP(J)-FLOW(IX)
GO TO 330
320 CHANGE=FLOW(IX)
IF(CHANGE.GE.DELTA)GO TO 340
DELTA=CHANGE
K=IX
340 IX=PRED(IX)
GO TO 310
350 J=LINK(IY)
IF(J.LT.0)GO TO 360
CHANGE=FLOW(IY)
GO TO 370
360 CHANGE=CAP(-J)-FLOW(IY)
IF(CHANGE.GE.DELTA)GO TO 380
DELTA=CHANGE
K=IX
LEG=-1
380 IY=PRED(IY)
GO TO 310
390 CONTINUE
IW=IX
DELTAI1=DELTA
IF(CODE.NE.5)GO TO 410
STATUS=STATPO
ENTER=PO
CMC+CMIN*STATUS*DELTA
DMD+DMIN*STATUS*DELTA
DIR=-1
IF(D1.GT.D)DIR=1
IF(DELTAI1.LT.ICAP)GO TO 400
CALL AUGMEN
GO TO 110
400 CALL UPDATE
GO TO 110
410 IF(DELTAI1.LE.MAX)GO TO 450
DELTA=MAX
ADD=CMIN*STATUS*DELTA
ADD1=DMIN*STATUS*DELTA
CMC+ADD
DMD+ADD1
CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,REFLAG
,C2,D2)
C=C+ADD
D=D-ADD1
IF(OVER.EQ.1)GO TO 880
CALL ADD
DMD+ADD1
IF(ASS10-D10.LE.S.OR.ABS(D-D2).LE.S).AND.CODE.GE.3)GO TO 420
IF(ASS10-D10.GE.S.AND.ABS(D-D2).GE.S)GO TO 420
GO TO 110
CODE=ENTER=-CAP(ENTER)

```

```

12689
12690 IF(CMAX.EQ.MAX)JDELTA=MAX
12691 IF(CDELTA.LT.0.OR.DELTA.GE.DELTA1)GO TO 440
12692 DELTA=DELTA-MAX
12693 ADD1=CMIN*STATUS*DELTA
12694 ADD2=DMIN*STATUS*DELTA
12695 C=C+ADD1
12696 D=D+ADD2
12697 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
12698 1,C2,D2)
12699 IF(COVER.EQ.1)GO TO 430
12700 DELTA=DELTA+MAX
12701 CALL AUGMEN
12702 GO TO 880
12703 430 DELTA=MAX
12704 ADD1=CMIN*STATUS*DELTA
12705 ADD2=DMIN*STATUS*DELTA
12706 CALL AUGMEN
12707 GO TO 880
12708 440 DELTA=DELTA1-MAX
12709 ADD1=CMIN*STATUS*DELTA
12710 ADD2=DMIN*STATUS*DELTA
12711 C=C+ADD1
12712 D=D+ADD2
12713 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
12714 1,C2,D2)
12715 IF(COVER.EQ.1.AND.(DELTA.EQ.(DELTA1-MAX)))GO TO 430
12716 IF(COVER.EQ.1)GO TO 880
12717 C=C+ADD1
12718 D=D+ADD2
12719 CALL AUGMEN
12720 IF(CLE.GE.0)CALL AUGMEN
12721 IF(CLE.NE.0)CALL UPDATE
12722 IF(CLE.EQ.4)GO TO 250
12723 460 CMIN
12724 DMIN
12725 TAIL
12726 HEAD
12727 DELTA
12728 IF(STATPO.EQ.-1)FLOWPO=CAP(P0)-DELTA
12729 **** THIS PART OF THE PROGRAM SELECTS AN ARC RS FROM A SET OF
12730 NON-BASIC ARCS FOR FLOW AUGMENTATION IN TWO CYCLES SIMULTANEOUSLY
12731 ****
12732 470 CDBE2=1
12733 480 RS=0
12734 DO 490 I=1,N
12735 SUM(I)=0.0
12736 IF(CR.GE.C1.AND.C.LE.C2)GO TO 880
12737 KTER=ITER+1
12738 DO 520 I=1,N
12739 START=POINT(I)
12740 FINISH=POINT(I+1)-1
12741 IF(START.GT.FINISH)GO TO 520
12742 DO 520 J=START,FINISH
12743 K=HEAD(J)
12744 IF(CAP(J).LT.0)GO TO 500
12745 CDBAR=COST(J)+PI(I)-PI(K)
12746 DBAR=BDUD(J)+PID(I)-PID(K)
12747 STATUS=1
12748 GO TO 530
12749 500 DBAR=PI(K)-PI(I)-COST(J)
12750 DBAR=PID(K)-PID(I)-BDUD(J)
12751 STATUS=-1
12752 IF(CBPO.EQ.0)GO TO 520
12753 IF(CBPO/DBAR.GE.0)GO TO 520
12754 IF((CBPO-(DBPO*CBAR)/DBAR).LE.0)GO TO 520
12755 RSEJ
12756 TAILRS=1
12757 HEADRS=1
12758 STATUSRS=STATUS
12759 RATIO=CBPO/DBAR
12760 CORR=1-CBPO/DBAR
12761 CORRS=1-CBPO/DBAR
12762 IF(CR.LT.RS.EQ.-1)FLDRS=CAP(RS)
12763 520 IF(CR.EQ.550)
12764 IF(CR.GE.2.AND.RS.EQ.0)GO TO 530
12765 IF(CR.GE.2.AND.RS.NE.0)GO TO 540
12766 IF(CR.LT.2.AND.RS.EQ.0)GO TO 550
12767 IF(CR.LT.2.AND.RS.NE.0)GO TO 560
12768 IF(CR.GE.2.AND.RS.EQ.0)GO TO 570
12769 IF(CR.GE.2.AND.RS.NE.0)GO TO 580
12770 IF(CR.LT.2.AND.RS.EQ.0)GO TO 590
12771 IF(CR.LT.2.AND.RS.NE.0)GO TO 600
12772 IF(CR.GE.2.AND.RS.EQ.0)GO TO 610
12773 IF(CR.GE.2.AND.RS.NE.0)GO TO 620
12774 IF(CR.LT.2.AND.RS.EQ.0)GO TO 630
12775 IF(CR.LT.2.AND.RS.NE.0)GO TO 640
12776 IF(CR.GE.2.AND.RS.EQ.0)GO TO 650
12777 IF(CR.GE.2.AND.RS.NE.0)GO TO 660
12778 IF(CR.LT.2.AND.RS.EQ.0)GO TO 670
12779 IF(CR.LT.2.AND.RS.NE.0)GO TO 680
12780 IF(CR.GE.2.AND.RS.EQ.0)GO TO 690
12781 IF(CR.GE.2.AND.RS.NE.0)GO TO 700
12782 IF(CR.LT.2.AND.RS.EQ.0)GO TO 710
12783 IF(CR.LT.2.AND.RS.NE.0)GO TO 720
12784 IF(CR.GE.2.AND.RS.EQ.0)GO TO 730
12785 IF(CR.GE.2.AND.RS.NE.0)GO TO 740
12786 IF(CR.LT.2.AND.RS.EQ.0)GO TO 750
12787 IF(CR.LT.2.AND.RS.NE.0)GO TO 760
12788 IF(CR.GE.2.AND.RS.EQ.0)GO TO 770
12789 IF(CR.GE.2.AND.RS.NE.0)GO TO 780
12790 IF(CR.LT.2.AND.RS.EQ.0)GO TO 790
12791 IF(CR.LT.2.AND.RS.NE.0)GO TO 800
12792 IF(CR.GE.2.AND.RS.EQ.0)GO TO 810
12793 IF(CR.GE.2.AND.RS.NE.0)GO TO 820
12794 IF(CR.LT.2.AND.RS.EQ.0)GO TO 830
12795 IF(CR.LT.2.AND.RS.NE.0)GO TO 840
12796 IF(CR.GE.2.AND.RS.EQ.0)GO TO 850
12797 IF(CR.GE.2.AND.RS.NE.0)GO TO 860
12798 IF(CR.LT.2.AND.RS.EQ.0)GO TO 870
12799 IF(CR.LT.2.AND.RS.NE.0)GO TO 880
12800 IF(CR.GE.2.AND.RS.EQ.0)GO TO 890
12801 IF(CR.GE.2.AND.RS.NE.0)GO TO 900
12802 IF(CR.LT.2.AND.RS.EQ.0)GO TO 910
12803 IF(CR.LT.2.AND.RS.NE.0)GO TO 920
12804 IF(CR.GE.2.AND.RS.EQ.0)GO TO 930
12805 IF(CR.GE.2.AND.RS.NE.0)GO TO 940
12806 IF(CR.LT.2.AND.RS.EQ.0)GO TO 950
12807 IF(CR.LT.2.AND.RS.NE.0)GO TO 960
12808 IF(CR.GE.2.AND.RS.EQ.0)GO TO 970
12809 IF(CR.GE.2.AND.RS.NE.0)GO TO 980
12810 IF(CR.LT.2.AND.RS.EQ.0)GO TO 990
12811 IF(CR.LT.2.AND.RS.NE.0)GO TO 1000
12812 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1010
12813 IF(CR.GE.2.AND.RS.NE.0)GO TO 1020
12814 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1030
12815 IF(CR.LT.2.AND.RS.NE.0)GO TO 1040
12816 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1050
12817 IF(CR.GE.2.AND.RS.NE.0)GO TO 1060
12818 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1070
12819 IF(CR.LT.2.AND.RS.NE.0)GO TO 1080
12820 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1090
12821 IF(CR.GE.2.AND.RS.NE.0)GO TO 1100
12822 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1110
12823 IF(CR.LT.2.AND.RS.NE.0)GO TO 1120
12824 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1130
12825 IF(CR.GE.2.AND.RS.NE.0)GO TO 1140
12826 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1150
12827 IF(CR.LT.2.AND.RS.NE.0)GO TO 1160
12828 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1170
12829 IF(CR.GE.2.AND.RS.NE.0)GO TO 1180
12830 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1190
12831 IF(CR.LT.2.AND.RS.NE.0)GO TO 1200
12832 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1210
12833 IF(CR.GE.2.AND.RS.NE.0)GO TO 1220
12834 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1230
12835 IF(CR.LT.2.AND.RS.NE.0)GO TO 1240
12836 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1250
12837 IF(CR.GE.2.AND.RS.NE.0)GO TO 1260
12838 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1270
12839 IF(CR.LT.2.AND.RS.NE.0)GO TO 1280
12840 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1290
12841 IF(CR.GE.2.AND.RS.NE.0)GO TO 1300
12842 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1310
12843 IF(CR.LT.2.AND.RS.NE.0)GO TO 1320
12844 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1330
12845 IF(CR.GE.2.AND.RS.NE.0)GO TO 1340
12846 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1350
12847 IF(CR.LT.2.AND.RS.NE.0)GO TO 1360
12848 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1370
12849 IF(CR.GE.2.AND.RS.NE.0)GO TO 1380
12850 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1390
12851 IF(CR.LT.2.AND.RS.NE.0)GO TO 1400
12852 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1410
12853 IF(CR.GE.2.AND.RS.NE.0)GO TO 1420
12854 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1430
12855 IF(CR.LT.2.AND.RS.NE.0)GO TO 1440
12856 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1450
12857 IF(CR.GE.2.AND.RS.NE.0)GO TO 1460
12858 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1470
12859 IF(CR.LT.2.AND.RS.NE.0)GO TO 1480
12860 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1490
12861 IF(CR.GE.2.AND.RS.NE.0)GO TO 1500
12862 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1510
12863 IF(CR.LT.2.AND.RS.NE.0)GO TO 1520
12864 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1530
12865 IF(CR.GE.2.AND.RS.NE.0)GO TO 1540
12866 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1550
12867 IF(CR.LT.2.AND.RS.NE.0)GO TO 1560
12868 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1570
12869 IF(CR.GE.2.AND.RS.NE.0)GO TO 1580
12870 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1590
12871 IF(CR.LT.2.AND.RS.NE.0)GO TO 1600
12872 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1610
12873 IF(CR.GE.2.AND.RS.NE.0)GO TO 1620
12874 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1630
12875 IF(CR.LT.2.AND.RS.NE.0)GO TO 1640
12876 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1650
12877 IF(CR.GE.2.AND.RS.NE.0)GO TO 1660
12878 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1670
12879 IF(CR.LT.2.AND.RS.NE.0)GO TO 1680
12880 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1690
12881 IF(CR.GE.2.AND.RS.NE.0)GO TO 1700
12882 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1710
12883 IF(CR.LT.2.AND.RS.NE.0)GO TO 1720
12884 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1730
12885 IF(CR.GE.2.AND.RS.NE.0)GO TO 1740
12886 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1750
12887 IF(CR.LT.2.AND.RS.NE.0)GO TO 1760
12888 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1770
12889 IF(CR.GE.2.AND.RS.NE.0)GO TO 1780
12890 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1790
12891 IF(CR.LT.2.AND.RS.NE.0)GO TO 1800
12892 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1810
12893 IF(CR.GE.2.AND.RS.NE.0)GO TO 1820
12894 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1830
12895 IF(CR.LT.2.AND.RS.NE.0)GO TO 1840
12896 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1850
12897 IF(CR.GE.2.AND.RS.NE.0)GO TO 1860
12898 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1870
12899 IF(CR.LT.2.AND.RS.NE.0)GO TO 1880
12900 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1890
12901 IF(CR.GE.2.AND.RS.NE.0)GO TO 1900
12902 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1910
12903 IF(CR.LT.2.AND.RS.NE.0)GO TO 1920
12904 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1930
12905 IF(CR.GE.2.AND.RS.NE.0)GO TO 1940
12906 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1950
12907 IF(CR.LT.2.AND.RS.NE.0)GO TO 1960
12908 IF(CR.GE.2.AND.RS.EQ.0)GO TO 1970
12909 IF(CR.GE.2.AND.RS.NE.0)GO TO 1980
12910 IF(CR.LT.2.AND.RS.EQ.0)GO TO 1990
12911 IF(CR.LT.2.AND.RS.NE.0)GO TO 2000
12912 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2010
12913 IF(CR.GE.2.AND.RS.NE.0)GO TO 2020
12914 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2030
12915 IF(CR.LT.2.AND.RS.NE.0)GO TO 2040
12916 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2050
12917 IF(CR.GE.2.AND.RS.NE.0)GO TO 2060
12918 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2070
12919 IF(CR.LT.2.AND.RS.NE.0)GO TO 2080
12920 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2090
12921 IF(CR.GE.2.AND.RS.NE.0)GO TO 2100
12922 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2110
12923 IF(CR.LT.2.AND.RS.NE.0)GO TO 2120
12924 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2130
12925 IF(CR.GE.2.AND.RS.NE.0)GO TO 2140
12926 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2150
12927 IF(CR.LT.2.AND.RS.NE.0)GO TO 2160
12928 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2170
12929 IF(CR.GE.2.AND.RS.NE.0)GO TO 2180
12930 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2190
12931 IF(CR.LT.2.AND.RS.NE.0)GO TO 2200
12932 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2210
12933 IF(CR.GE.2.AND.RS.NE.0)GO TO 2220
12934 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2230
12935 IF(CR.LT.2.AND.RS.NE.0)GO TO 2240
12936 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2250
12937 IF(CR.GE.2.AND.RS.NE.0)GO TO 2260
12938 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2270
12939 IF(CR.LT.2.AND.RS.NE.0)GO TO 2280
12940 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2290
12941 IF(CR.GE.2.AND.RS.NE.0)GO TO 2300
12942 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2310
12943 IF(CR.LT.2.AND.RS.NE.0)GO TO 2320
12944 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2330
12945 IF(CR.GE.2.AND.RS.NE.0)GO TO 2340
12946 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2350
12947 IF(CR.LT.2.AND.RS.NE.0)GO TO 2360
12948 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2370
12949 IF(CR.GE.2.AND.RS.NE.0)GO TO 2380
12950 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2390
12951 IF(CR.LT.2.AND.RS.NE.0)GO TO 2400
12952 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2410
12953 IF(CR.GE.2.AND.RS.NE.0)GO TO 2420
12954 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2430
12955 IF(CR.LT.2.AND.RS.NE.0)GO TO 2440
12956 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2450
12957 IF(CR.GE.2.AND.RS.NE.0)GO TO 2460
12958 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2470
12959 IF(CR.LT.2.AND.RS.NE.0)GO TO 2480
12960 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2490
12961 IF(CR.GE.2.AND.RS.NE.0)GO TO 2500
12962 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2510
12963 IF(CR.LT.2.AND.RS.NE.0)GO TO 2520
12964 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2530
12965 IF(CR.GE.2.AND.RS.NE.0)GO TO 2540
12966 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2550
12967 IF(CR.LT.2.AND.RS.NE.0)GO TO 2560
12968 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2570
12969 IF(CR.GE.2.AND.RS.NE.0)GO TO 2580
12970 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2590
12971 IF(CR.LT.2.AND.RS.NE.0)GO TO 2600
12972 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2610
12973 IF(CR.GE.2.AND.RS.NE.0)GO TO 2620
12974 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2630
12975 IF(CR.LT.2.AND.RS.NE.0)GO TO 2640
12976 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2650
12977 IF(CR.GE.2.AND.RS.NE.0)GO TO 2660
12978 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2670
12979 IF(CR.LT.2.AND.RS.NE.0)GO TO 2680
12980 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2690
12981 IF(CR.GE.2.AND.RS.NE.0)GO TO 2700
12982 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2710
12983 IF(CR.LT.2.AND.RS.NE.0)GO TO 2720
12984 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2730
12985 IF(CR.GE.2.AND.RS.NE.0)GO TO 2740
12986 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2750
12987 IF(CR.LT.2.AND.RS.NE.0)GO TO 2760
12988 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2770
12989 IF(CR.GE.2.AND.RS.NE.0)GO TO 2780
12990 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2790
12991 IF(CR.LT.2.AND.RS.NE.0)GO TO 2800
12992 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2810
12993 IF(CR.GE.2.AND.RS.NE.0)GO TO 2820
12994 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2830
12995 IF(CR.LT.2.AND.RS.NE.0)GO TO 2840
12996 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2850
12997 IF(CR.GE.2.AND.RS.NE.0)GO TO 2860
12998 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2870
12999 IF(CR.LT.2.AND.RS.NE.0)GO TO 2880
13000 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2890
13001 IF(CR.GE.2.AND.RS.NE.0)GO TO 2900
13002 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2910
13003 IF(CR.LT.2.AND.RS.NE.0)GO TO 2920
13004 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2930
13005 IF(CR.GE.2.AND.RS.NE.0)GO TO 2940
13006 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2950
13007 IF(CR.LT.2.AND.RS.NE.0)GO TO 2960
13008 IF(CR.GE.2.AND.RS.EQ.0)GO TO 2970
13009 IF(CR.GE.2.AND.RS.NE.0)GO TO 2980
13010 IF(CR.LT.2.AND.RS.EQ.0)GO TO 2990
13011 IF(CR.LT.2.AND.RS.NE.0)GO TO 3000
13012 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3010
13013 IF(CR.GE.2.AND.RS.NE.0)GO TO 3020
13014 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3030
13015 IF(CR.LT.2.AND.RS.NE.0)GO TO 3040
13016 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3050
13017 IF(CR.GE.2.AND.RS.NE.0)GO TO 3060
13018 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3070
13019 IF(CR.LT.2.AND.RS.NE.0)GO TO 3080
13020 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3090
13021 IF(CR.GE.2.AND.RS.NE.0)GO TO 3100
13022 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3110
13023 IF(CR.LT.2.AND.RS.NE.0)GO TO 3120
13024 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3130
13025 IF(CR.GE.2.AND.RS.NE.0)GO TO 3140
13026 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3150
13027 IF(CR.LT.2.AND.RS.NE.0)GO TO 3160
13028 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3170
13029 IF(CR.GE.2.AND.RS.NE.0)GO TO 3180
13030 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3190
13031 IF(CR.LT.2.AND.RS.NE.0)GO TO 3200
13032 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3210
13033 IF(CR.GE.2.AND.RS.NE.0)GO TO 3220
13034 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3230
13035 IF(CR.LT.2.AND.RS.NE.0)GO TO 3240
13036 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3250
13037 IF(CR.GE.2.AND.RS.NE.0)GO TO 3260
13038 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3270
13039 IF(CR.LT.2.AND.RS.NE.0)GO TO 3280
13040 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3290
13041 IF(CR.GE.2.AND.RS.NE.0)GO TO 3300
13042 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3310
13043 IF(CR.LT.2.AND.RS.NE.0)GO TO 3320
13044 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3330
13045 IF(CR.GE.2.AND.RS.NE.0)GO TO 3340
13046 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3350
13047 IF(CR.LT.2.AND.RS.NE.0)GO TO 3360
13048 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3370
13049 IF(CR.GE.2.AND.RS.NE.0)GO TO 3380
13050 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3390
13051 IF(CR.LT.2.AND.RS.NE.0)GO TO 3400
13052 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3410
13053 IF(CR.GE.2.AND.RS.NE.0)GO TO 3420
13054 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3430
13055 IF(CR.LT.2.AND.RS.NE.0)GO TO 3440
13056 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3450
13057 IF(CR.GE.2.AND.RS.NE.0)GO TO 3460
13058 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3470
13059 IF(CR.LT.2.AND.RS.NE.0)GO TO 3480
13060 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3490
13061 IF(CR.GE.2.AND.RS.NE.0)GO TO 3500
13062 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3510
13063 IF(CR.LT.2.AND.RS.NE.0)GO TO 3520
13064 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3530
13065 IF(CR.GE.2.AND.RS.NE.0)GO TO 3540
13066 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3550
13067 IF(CR.LT.2.AND.RS.NE.0)GO TO 3560
13068 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3570
13069 IF(CR.GE.2.AND.RS.NE.0)GO TO 3580
13070 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3590
13071 IF(CR.LT.2.AND.RS.NE.0)GO TO 3600
13072 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3610
13073 IF(CR.GE.2.AND.RS.NE.0)GO TO 3620
13074 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3630
13075 IF(CR.LT.2.AND.RS.NE.0)GO TO 3640
13076 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3650
13077 IF(CR.GE.2.AND.RS.NE.0)GO TO 3660
13078 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3670
13079 IF(CR.LT.2.AND.RS.NE.0)GO TO 3680
13080 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3690
13081 IF(CR.GE.2.AND.RS.NE.0)GO TO 3700
13082 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3710
13083 IF(CR.LT.2.AND.RS.NE.0)GO TO 3720
13084 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3730
13085 IF(CR.GE.2.AND.RS.NE.0)GO TO 3740
13086 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3750
13087 IF(CR.LT.2.AND.RS.NE.0)GO TO 3760
13088 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3770
13089 IF(CR.GE.2.AND.RS.NE.0)GO TO 3780
13090 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3790
13091 IF(CR.LT.2.AND.RS.NE.0)GO TO 3800
13092 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3810
13093 IF(CR.GE.2.AND.RS.NE.0)GO TO 3820
13094 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3830
13095 IF(CR.LT.2.AND.RS.NE.0)GO TO 3840
13096 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3850
13097 IF(CR.GE.2.AND.RS.NE.0)GO TO 3860
13098 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3870
13099 IF(CR.LT.2.AND.RS.NE.0)GO TO 3880
13100 IF(CR.GE.2.AND.RS.EQ.0)GO TO 3890
13101 IF(CR.GE.2.AND.RS.NE.0)GO TO 3900
13102 IF(CR.LT.2.AND.RS.EQ.0)GO TO 3910
13103 IF(CR.LT.2.AND.RS.NE.0)GO TO 3920
13104 IF(CR.GE.2.AND.RS.EQ.0)
```

```

1 IF(CBPO.GT.0)GO TO 540
2 CBPO=-CBPO
3 DBPO=-DBPO
4 CAP(PQ)=CAP(PQ)
540 IF(DBPO.LT.0.AND.DIR.EQ.-1)RFLAG=-1
6 IF(DBPO.GT.0.AND.DIR.EQ.1)RFLAG=+1
7 IF(RFLAG.NE.0)SIGN=-SIGN
8 FLAG=1
9 FLOW=FLOWPQ
10 GO TO 240
11 **** THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE
12 MINIMUM FLOW TO BE AUGMENTED IN THE TWO CYCLES SIMULTANEOUSLY
13 ****
14 550 CODE1=1
15 DELTA1=FLOWPQ
16 IF(CSTATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
17 LEG=0
18 DELTA2=IABS(CAP(RS))/RATIO
19 IF(DELTA2.LT.DELTA1)DELTA1=DELTA2
20 ENTERPQ
21 STATUS=STATPQ
22 RINC=1
23 INC=1
24 ITAIL=TAILPO
25 IHEAD=HEADPO
26 570 IF(STATUS.EQ.-1)GO TO 580
27 IX=ITAIL
28 IY=IHEAD
29 GO TO 590
30 580 IX=IHEAD
31 IY=ITAIL
32 590 IF(IX.EQ.IY)GO TO 750
33 IF(NUMBER(IX).GT.NUMBER(IY))GO TO 670
34 IF(CODE1.EQ.3)GO TO 620
35 J=LINK(IX)
36 IF(J.LT.0)GO TO 600
37 SUM(IX)=SUM(IX)+RIN
38 GO TO 610
39 600 SUM(IX)=SUM(IX)-RIN
40 610 IF(CODE1.EQ.1)GO TO 660
41 620 IF(SUM(IX).GT.630,660,640)
42 630 DELTA2=-FLOW(IX)/SUM(IX)
43 GO TO 650
44 640 J=IABS(LINK(IX))
45 DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
46 650 IF(DELTA2.GE.DELTA1) GO TO 660
47 DELTA1=DELTA2
48 LEG=INC
49 K1=IX
50 TOTAL=FLOW(K1)+DELTA1*SUM(K1)
51 660 IX=PRED(IX)
52 GO TO 590
53 670 IF(CODE1.GT.2)GO TO 700
54 J=LINK(IY)
55 IF(J.LT.0)GO TO 680
56 SUM(IY)=SUM(IY)-RIN
57 GO TO 690
58 680 SUM(IY)=SUM(IY)+RIN
59 690 IF(CODE1.EQ.1)GO TO 740
60 700 IF(SUM(IY).GT.710,740,720)
61 710 DELTA2=-FLOW(IY)/SUM(IY)
62 GO TO 730
63 720 J=IABS(LINK(IY))
64 DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
65 730 IF(DELTA2.GE.DELTA1)GO TO 740
66 DELTA1=DELTA2
67 LEG=INC
68 K1=IY
69 TOTAL=FLOW(K1)+DELTA1*SUM(K1)
70 740 IY=PRED(IY)
71 GO TO 590
72 750 IF(CODE1.EQ.1)I1=IX
73 IF(CODE1.EQ.2)I2=IX
74 CODE1=CODE1+1
75 IF(CODE1.GT.3)GO TO 760
76 IF(CODE1.EQ.3)GO TO 560
77 ENDERS
78 ITAIL=TAILRS
79 IHEAD=HEADRS
80 STATUS=STATRS
81 RINC=RATIO
82 RINC=2
83 GO TO 570
84 760 DELTA2=(C1-C)/(CBPO+RATIO*C8RS)
85 IF(C1.LT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
86 C=C-(DELTA1-CBPO+RATIO*DELTA1)*C8RS
87 DELTA2=(C1-C)/(CBPO+RATIO*DELTA1+C8RS)

```



```

DELTIA=DELTIA
IWEIN1
IF(PO>FLOWPO+STATPO*DELTIA1
CALL ADJGEN
ITAILE=TAILRS
IHEAD=HEADRS
STATUS=STATRS
DELTIA=DELTIA1*RATIO
IN=IN2
LEG=LEG/2
ENTER=RS
K=K1
DMIN=CBRS
DMIN=D8RS
IF(STATUS.EQ.1)GO TO 850
DMIN=DMIN
DMIN=DMIN
ICAP=IABS(CAP(RS))
850 RECDN=FLOWRS
CALL UPDATE
CAP(LEAVE)=-IABS(CAP(LEAVE))
IF(COTAL LE,S)CAP(LEAVE)=-CAP(LEAVE)
IF(COTAL PO+PI(TAILPO)-PI(HEADPO)
DBPO=COST(PO)+PI(TAILPO)-PI(HEADPO)
DBPO=BUD(PO)+PI(PO)-PI(HEADPO)
IF(STATPO.EQ.1)GO TO 860
IF(STATPO.EQ.1)GO TO 860
CBPO=-CBPO
DBPO=-DBPO
860 STATPO=-STATPO
CBPO=-CBPO
DBPO=-DBPO
870 GO TO 770
880 CALL RTIME(TIME2)
TIME=TIME2-TIME1
WRITE(22,890),C,D,TIME,ITER
FORMAT(1X,'COST=',F14.4,'TIME=',I10,'ITER=',I8)
890 FORMAT(1X,'BUDGET=',F14.4)
STOP
END

```

```

* THIS SUBROUTINE UPDATES THE OBJECTIVE FUNCTION OF INTERVAL.
* GOAL PROGRAMMING
* SUBROUTINE UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,
1      RFLAG,C2,D2)
REAL *16,MINIM
INTEGER OVER
IF(DELTA.EQ.0)GO TO 50
IF(C.GT.C2.OR.C.LT.C1.OR.D.LT.D1.OR.D.GT.D2)GO TO 10
OVER=1
GO TO 50
10 IF(C.GT.C1.AND.C.LT.C2)GO TO 50
IF(D.GT.D1.AND.D.LT.D2)GO TO 50
IF(RFLAG)70,20,30
20 IF(DIR)30,50,70
30 IF(C.GE.C2)GO TO 60
MINIM=W1*(C1-C)+W2*(D-D2)
40 OVER=1
IF(MIN.GE.MINIM)GO TO 50
MINIM=MIN
OVER=0
50 TYPE*,C,D,C1,D1,C2,D2,MIN,MINIM,OVER
RETURN
60 MIN=W3*(C-C2)+W4*(D-D2)
GO TO 40
70 IF(C.GE.C2)GO TO 80
MIN=W1*(C1-C)+W2*(D1-D)
GO TO 40
80 MIN=W3*(C-C2)+W2*(D1-D)
GO TO 40
END

```

```

***** THIS SUBROUTINE UPDATES THE DUAL VARIABLES PI'S AND PID'S, ****
* THREAD INDICES, NO. OF SUCCESSORS, PREDECESSOR INDICES AND ****
* FLOW ****
***** SUBROUTINE UPDATE ****
      INTEGER WIDTH, SEED, ROOT, SOURCE, SINK, PREV, COUNT, FRONT, START,
      FINISH, ENTER, V, STATUS, QP, Z, X, R,
      TIME1, TIME2, TIME, CMIN, TR,
      PRED(405), THREAD(405), PI(405), POINT(405), PID(0:405),
      SCAN(405), HEAD(3000), CAP(3000), CUST(3000), BUD(0:3000),
      P0, RS, TAILPO, TAILRS, HEADPO, HEADRS, CODE, CODE1, CODE2
      REAL MAX, MAX1, MAX2, MUF, MUEMAX, ICAP, IFLOW, JFLOW
      DIMENSION NUMBER(405), LINK(405), LIST(405), FLOW(405), SUM(405)
      COMMON ITAIL, IHEAD, STATUS, DELTA, ICAP, LEG, ENTER, CMIN, DMIN, IW,
      LINK, FLOW, PRED, SUM, CAP, NUMBER, THREAD, PI, PID, IFLOW, K
      LEAVE, RFLOW
      C
      IFLOW=RFLOW+STATUS*DELTA
      IF(STATUS.EQ.-1)IFLOW=ICAP-DELTA
      ILINK=-STATUS*LEG*ENTER
      IF(ILINK.GT.0)GO TO 610
      O=ITAIL
      OOP=IHEAD
      DMIN=-CMIN
      DMIN=-DMIN
      GO TO 620
  610  O=IHEAD
      OOP=ITAIL
      CHANGE=LEG*DELTA
      KP=PRED(K)
      LEAVE=IABS(LINK(K))
      KODE=GEG
      IF(LINK(K).LT.0)KODE=-KODE
      IF(KODE.EQ.1)CAP(LEAVE)=-CAP(LEAVE)
      CAP(CENTER)=ICAP
      IPRED=QP
      INUMB=NUMBER(K)
      L=0
      I=0
  625  JFLOW=FLOW(I)
      JLINK=LINK(I)
      JPRED=PRED(I)
      JSUCC=NUMBER(I)-L
      Z=I
      X=THREAD(I)
      COUNT=2
  630  IF(COUNT.GT.JSUCC)GO TO 640
      PI(X)=PI(X)+CMIN
      PID(X)=PID(X)+DMIN
      Z=X
      X=THREAD(X)
      COUNT=COUNT+1
      GO TO 630
  640  R=JPRED
  650  TR=THREAD(R)
      IF(TR.EQ.I)GO TO 660
      R=TR
      GO TO 650
  660  FLOW(I)=IFLOW
      LINK(I)=ILINK
      NUMBER(I)=INUMB
      PRED(I)=IPRED
      PI(I)=PI(I)+CMIN
      PID(I)=PID(I)+DMIN
      THREAD(R)=K
      IF(I.EQ.K)GO TO 690
      THREAD(Z)=JPRED
      IF(JLINK.LT.0)GO TO 670
      IFLOW=JFLOW+CHANGE
      GO TO 680
  670  IFLOW=JFLOW-CHANGE
      ILINK=-JLINK
      INUMB=INUMB-JSUCC
      D=L+JSUCC
      IPRED=I
      I=JPRED
      GO TO 625
  680  THREAD(Z)=THREAD(O)
      THREAD(O)=0
      CHANGE=-CHANGE
      NUMBER=NUMBER(0)
      I=QP
  710  IF(I.EQ.1)GO TO 740
      I=1
      IF(JLINK(I).GE.0)GO TO 720
      FLOW(I)=FLOW(I)+CHANGE
      INUMB(I)=NUMBER(I)+NUMBER
      I=PRED(I)

```

1930 720 FLOW(I)=FLOW(I)-CHANGE
1931 720 FLOW(I)=FLOW(I)+NUMBER(I)
1932 720 I=PRED(I)
1933 720 GO TO 710
1934 740 I=KP
1935 740 CHANGE==CHANGE
1936 740 IF(I.EQ.IW)RETURN
1937 740 IF(LINK(I).LT.0)GO TO 760
1938 740 FLOW(I)=FLOW(I)+CHANGE
1939 740 NUMBER(I)=NUMBER(I)-NUMBER
1940 740 I=PRED(I)
1941 740 GO TO 750
1942 740 FLOW(I)=FLOW(I)-CHANGE
1943 740 NUMBER(I)=NUMBER(I)-NUMBER
1944 740 I=PRED(I)
1945 740 GO TO 750
1946 740
1947 750 END

```

0001
0002 * THIS SUBROUTINE UPDATES THE FLOW
0003
0004 SUBROUTINE AUGMEN
0005      INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
0006      1      FINISH,ENTER,V,STATUS,0,QP,Z,X,R,
0007      2      TIME1,TIME2,TIME,CMIN,TR,
0008      3      PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
0009      4      SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
0010      5      PQ,RS,TAILPO,TAILRS,HEADPO,HEADRS,CJOE,CJOE1,CJOE2
0011      REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,IJFLOW
0012      DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
0013      COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
0014      1      LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
0015      2      LEAVE,RFLOW
0016      CAP(ENTER)=-CAP(ENTER)
0017      CHANGE=DELTA*STATUS
0018      I=ITAIL
0019      410     IF(I.EQ.IW)GO TO 440
0020      J=LINK(I)
0021      IF(J.LT.0)GO TO 420
0022      FLOWN(I)=FLOW(I)+CHANGE
0023      I=PRED(I)
0024      GO TO 410
0025      420     FLOWN(I)=FLOW(I)-CHANGE
0026      I=PRED(I)
0027      GO TO 410
0028      440     I=IHEAD
0029      450     IF(I.EQ.IW)RETURN
0030      J=LINK(I)
0031      IF(J.LT.0)GO TO 460
0032      FLOWN(I)=FLOW(I)-CHANGE
0033      I=PRED(I)
0034      GO TO 450
0035      460     FLOWN(I)=FLOW(I)+CHANGE
0036      I=PRED(I)
0037      GO TO 450
0038      END

```

```

THIS PART OF THE PROGRAM ( SURROUTINES NGA , STAR AND
THIS FUNCTION IRAN ) GENERATES THE NETWORK
SUBROUTINE NGA(WIDTH, LENGTH, N, M, S, T, POINT, HEAD)
* THIS SUBROUTINE GENERATES THE HEAD NODES OF ARCS
INTEGER S,T,COUNT,WIDTH
      ALIST(3000),BLIST(3000),POINT(405),RPOINT(405),
      TAIL(3000),HEAD(3000),CAP(3000),TRACE(3000)
      LARGE=100000
      N=WIDTH*WIDTH+2
      S=1
      T=N
      IA=LARGE
      IR=LARGE
      IC=0
      LD=100
      IW=2
      IX=WIDTH+1
      IY=N-WIDTH
      IZ=N-1
      DO 10 I=1,N
      ALIST(I)=0
      CONTINUE
      COUNT=WIDTH*(LENGTH+1)+1
      DO 20 J=COUNT,M
      ITAIL=IRAN(1,N-1)
      ALIST(ITAIL)=ALIST(ITAIL)+1
      20
      CONTINUE
      COUNT=0
      DO 30 I=IW,IX-1
      COUNT=COUNT+1
      TAIL(COUNT)=I
      HEAD(COUNT)=I
      CAP(COUNT)=IRAN(IA,IB)
      30
      CONTINUE
      DO 40 K=1,IZ
      DO 50 I=1,N
      BLIST(I)=0
      I1=MIN0(K+WIDTH,N)
      BLIST(I1)=1
      IF(BLIST(K).EQ.0)GO TO 110
      DO 100 J=1,ALIST(K)
      IHEAD=IRAN(2,N)
      IF(BLIST(IHEAD).EQ.1)GO TO 90
      IF(IHEAD.EQ.K)GO TO 90
      BLIST(IHEAD)=1
      90
      CONTINUE
      DO 110 I=N,2,-1
      IF(BLIST(I).EQ.0)GO TO 120
      COUNT=COUNT+1
      TAIL(COUNT)=K
      HEAD(COUNT)=I
      IF(TAIL(COUNT).EQ.1.AND.HEAD(COUNT).EQ.IX)GO TO 130
      IF(HEAD(COUNT).EQ.N.AND.HEAD(COUNT)=TAIL(COUNT).LE.WIDTH)
      1
      GO TO 130
      1
      CAP(COUNT)=IRAN(IC,TD)
      GO TO 120
      1
      IF(CAP(COUNT)=IRAN(IA,IB)
      1
      CONTINUE
      1
      CONTINUE
      1
      CALL STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)
      1
      RETURN
      END

```

```

0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038

THIS SUBROUTINE GENERATES POINTERS OF THE NETWORK
SUBROUTINE STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)
INTEGER COUNT,POINT(405),RPOINT(405),ALIST(3000),BLIST(3000),
1 TAIL(3000),HEAD(3000),TRACE(3000),CAP(3000)

DO 10 I=1,N
ALIST(I)=0
CONTINUE
POINT(1)=1
K=1
DO 20 J=1,M
ALIST(HEAD(J))=ALIST(HEAD(J))+1
IF(TAIL(J).EQ.K)GO TO 20
K=K+1
POINT(K)=J
CONTINUE
POINT(N)=M+1
POINT(N+1)=M+1
CALL SORT(N,M,POINT,TAIL,HEAD,CAP)
BLIST(1)=1
RPOINT(1)=1
DO 30 I=2,N
BLIST(I)=BLIST(I-1)+ALIST(I-1)
RPOINT(I)=BLIST(I)
CONTINUE
RPOINT(N+1)=M+1
DO 40 J=M,1,-1
JHEAD=HEAD(J)
TRACE(BLIST(JHEAD))=J
BLIST(JHEAD)=BLIST(JHEAD)+1
CONTINUE
RETURN
END

```

THIS FUNCTION GENERATES A RANDOM NUMBER BETWEEN IC AND ID.

FUNCTION IRAN(IC, ID)
IRAN=IC+(ID-IC+1)*RAN(X)
IF (IRAN.GT.ID) IRAN=ID
RETURN
END

A 87470